DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-5002

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

N2452

DEVELOPMENT OF A HYPERTEXT ORIENTED TECHNICAL INFORMATION MANAGEMENT SYSTEM.

by

Peter Andrew Nardi

December 1989

Thesis Advisor: C. Thomas Wu

Approved for public release; distribution is unlimited

T247295

Unclassified
Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification UNCLASSIFIED
1b Restrictive Markings
2a Security Classification Authority
3 Distribution Availability of Report
2b Declassification/Downgrading Schedule
Approved for public release; distribution is unlimited.

4 Performing Organization Report Number(s)
5 Monitoring Organization Report Number(s)
6a Name of Performing Organization: Naval Postgraduate School
6b Office Symbol (If Applicable): 37
7a Name of Monitoring Organization: Naval Postgraduate School
6c Address (city, state, and ZIP code): Monterey, CA 93943-5000
7b Address (city, state, and ZIP code): Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization
8b Office Symbol (If Applicable)
9 Procurement Instrument Identification Number
8c Address (city, state, and ZIP code)
10 Source of Funding Numbers

Program Element Number	Project No	Task No	Work Unit Accession No

11 Title (Include Security Classification) DEVELOPMENT OF A HYPERTEXT ORIENTED TECHNICAL INFORMATION MANAGEMENT SYSTEM.

12 Personal Author(s) Nardi, Peter, A.

13a Type of Report	13b Time Covered From	To	14 Date of Report (year, month,day)	15 Page Count
Master's Thesis			December 1989	96

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes

Field	Group	Subgroup

18 Subject Terms (continue on reverse if necessary and identify by block number)
Text Searching, Database, HyperText

19 Abstract (continue on reverse if necessary and identify by block number

Argos is a prototype multi-media database system being developed, utilizing modern off the shelf technology, to produce a new generation of shipboard data management tools. The ultimate goal of the project is to provide database support for the "Paperless Ship" Navy. This work extends Argos' functionality by providing on-line access to technical information, and the tools necessary to conduct key word searches of large volumes of textual information.

20 Distribution/Availability of Abstract
[X] unclassified/unlimited [ ] same as report [ ] DTIC users
21 Abstract Security Classification
Unclassified

22a Name of Responsible Individual: C.T. Wu
22b Telephone (Include Area code): (408) 646-3391
22c Office Symbol: 52Wq

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete
security classification of this page
Unclassified

Approved for public release; distribution is unlimited.

Development of a HyperText Oriented Technical Information Management System

by

Peter Andrew Nardi
Lieutenant, United States Navy
B.S., United States Naval Academy, 1984

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1989

ABSTRACT

Argos is a prototype multi-media database system being developed, utilizing modern off the shelf technology, to produce a new generation of shipboard data management tools. The ultimate goal of the project is to provide database support for the "Paperless Ship" Navy. This work extends Argos' functionality by providing on-line access to technical information, and the tools necessary to conduct key word searches of large volumes of textual information.

T1013
N2452
c.1

TABLE OF CONTENTS

I. INTRODUCTION 1

II. THE PROBLEM STATEMENT 6

A. BACKGROUND 6

B. PROBLEMS TIM SHOULD ADDRESS 7

1. Quantity of Information Provided 7

2. Type of Information Provided 8

3. Converting Printed Material 9

4. User Interface Issues 10

C. CONCLUSION 10

III. DEVELOPMENT ENVIRONMENT (HYPERCARD) 12

A. WHY HYPERCARD WAS CHOSEN TO DEVELOP TIM 12

B. FACILITIES PROVIDED BY HYPERCARD 13

1. Stacks, Backgrounds, Cards, Fields and Buttons 13

2. Graphic Design Tools 14

3. HyperTalk and Scripting 15

4. XCMDs and XFCNs 16

C. HYPERCARD COMPARED TO OBJECT ORIENTED PROGRAMMING (OOP) LANGUAGES 16

1. Object Oriented (OOP) Programming Languages 16

2. Why HyperCard is not a True OOP Language 18

D. ADVANCEMENTS IN HYPERCARD 20

IV. DESIGN AND IMPLEMENTATION 22

A. TECHNICAL MANUAL CONVERSION 22

1. From Printed to Electronic Form 22

2. Index Stack 22

3. Text Stack 24

B. SEARCHING FOR INFORMATION 27

1. The Search Algorithm 27

2. Defining a Search 30
3. Choosing From the Results of a Search - The Customized Table of Contents 33
4. Managing the Customized Table of Contents 36
5. Transition From the Customized Table of Contents to the Text Stack 36
6. The Function of the Various Buttons 37
C. SYSTEM MAINTENANCE 37
1. Adding and Deleting Manuals 37
2. Updates and Routine Maintenance 38
V. CONCLUSION 40
A. SUMMARY OF RESEARCH 40
B. REVIEW OF RESULTS 41
C. PROPOSED EXTENSIONS AND IMPROVEMENTS 41

LIST OF FIGURES

Figure 1 Typical Stack Layout 13
Figure 2 HyperCard's Graphic Tool Menus 15
Figure 3 Inheritance 17
Figure 4 Hierarchy of Message Travel in HyperCard 19
Figure 5 Index Stack Arrangement 24
Figure 6 Text Page Layout 25
Figure 7 Opening TIM Window 30
Figure 8 Library Selection 31
Figure 9 Technical Manual Title Selection 31
Figure 10 Search Key Selection 33
Figure 11 Search Type Selection 34
Figure 12 Customized Table of Contents Card 35
Figure 13 System Manager Window 31

ACKNOWLEDGEMENTS

I would like to acknowledge the contributions of the following people who made this work possible:

- My thesis advisor, C. Thomas Wu, whose guidance and advice was greatly appreciated.
- Henry Turner, and Dionisis Antonopolous for their help with the style sheets and thesis formatting. Also for allowing me to use a large portion of their bibliography.
- Mark Scherfling for his *printfield* XCMD. Portions of which are copyrighted by Think Technologies, Inc.
- Andrew Gilmartin for his *NewFileName* and *FileName* XCMDs.
- Guy de Picciotto for his *Hpopupmenu* XCMD, version 1.1
- Eric Alderman for his *Script Report* stack version 1.1 which was used to print the source code for TIM.
- Steve Drazga for his *Developers Stack*, version 1.2r
- Last, but certainly not least, my loving wife Jeanne. Her unending support and confidence in me was there when I needed it the most.

I. INTRODUCTION

Today's modern navy is faced with a paperwork dilemma. As ships and the missions they perform become more complex there has been a dramatic increase in the amount of paperwork required to carry out those missions. From the administrative documentation of daily activities to the maintenance of personnel records, a whole array of shipboard activities which have traditionally relied on paperwork generation are quickly becoming too difficult to manage. Even shipboard functions which are not thought of as being paperwork intensive are not immune to this burden. It seems that each new weapon or engineering system placed on a ship brings with it a tremendous overhead in the form of Manufacture's Technical Manuals, Naval Ships Technical Manuals (NSTM's) and required logs/maintenance records. As the technological revolution progresses and computers become smaller, faster, more powerful, and less expensive it seems that there should be a solution to the paperwork dilemma on the horizon.

There have been some attempts at solutions to the problem and there are some interesting prospects. The Computer Aided Logistics Supply (CALS) system which is slated for insertion into the SSN-21 Attack Submarine Acquisition Program is an attempt to achieve increased productivity through data digitization and electronic storage of information [Ref 3]. However since CALS is not yet in production and will almost certainly go through some early difficulties that any new system experiences, it is too soon to judge if CALS will be effective or well received in the fleet.

One system that is currently in place and can provide a case study is the Shipboard Naval Automated Data Processing (SNAP II) system. This is a text

oriented data management tool designed to handle the routine supply and administrative functions encountered on board ship. There are also facilities to schedule and track preventative maintenance. However, the response to the SNAP system has been less than enthusiastic to the point where there is already a call for the next generation of, or a replacement for it. The biggest draw back in SNAP is that the current hardware in place is not suitable for implementing more advanced software with a better interface. To make SNAP II more useable not only means adding more capabilities, it means making it easier to use the capabilities it already has. This ease of use issue will be a major consideration for the next generation of shipboard computers.

One solution to the problem is taking shape as a series of student Master's thesis at the Naval Postgraduate School. The project known as "ARGOS" was originally introduced by CDR B. B. Giannotti, USN and Lt Kevin F. Duffy, USN and was intended to reduce the amount of printed material processed and generated on board modern Naval Vessels [Ref 1]. Their thesis developed the concept of a graphical interface to conventional text information management, along with the integration of sound to allow the user to navigate through the system more naturally. The programming environment they chose is known as HyperCard and runs on the Apple Macintosh family of computers.[1] In addition to providing the tools necessary to create easy to use graphical applications, the HyperCard environment also significantly reduces the amount time required to produce a working model.

---

[1]HyperCard and Macintosh are registered trademarks of Apple Computer Corporation.

HyperCard is somewhat unique in terms of a programming environment. While it takes an Object Oriented approach to applications programming, it is not a true Object Oriented Programming (OOP) language. Many of the constructs and capabilities that are present in traditional OOP languages like SmallTalk, C++, and Actor (an OOP language for the IBM pc) are missing from HyperCard. For example there are no user defined classes in HyperCard, only five system defined classes which are: Stacks, Backgrounds, Cards, Fields and Buttons. The programmer has only these to choose from when developing an application. HyperCard does, however, provide for extensions of its functionality through the use of a variety of built-in capabilities. These capabilities will be outlined in a more detailed description of HyperCard in Chapter 3.

The original project started by Giannotti and Duffy [Ref 1] set up the basic guidelines for the development of a paperless ship system. Along with their model they proposed a series of extensions to serve as follow on thesis work. One such extension involves the conversion of hard copy Naval Ship's Technical Manuals and Manufacture's Technical Publications into an electronic format, along with the development of a tool which will allow for fast and easy retrieval of desired information. The prototype of this tool, called the Technical Information Manager (TIM) forms the basis of this thesis. It was developed in the same programming environment as ARGOS (HyperCard) and models Volume One of the Engineering Propulsion Plant Manual for FFG-7 class ships. Like the rest of ARGOS, TIM is a multi-media (text, graphics and sound) environment and represents a break from traditional database approaches to information management.

Since the programming environment for TIM is dictated by ARGOS, the next step was to decide how best to represent paper based information in a computer to

allow for feasible information retrieval. The problem of accessing volumes of free form text is distinctly different from that of accessing a conventional database. In a conventional system the user constructs a query in one of several system specific languages. The database is arranged as a series of "records" with each record having "fields." Certain fields act as keys and by comparing only the value of these key fields with the user's query the desired results may be obtained.

In free form text searches the idea of keys takes on a different meaning. The keys are now those words or parts of words that the user wants to see information about, and in effect every word of the volume being searched becomes a key. What the user wants the system to do is search an entire technical manual and return some information about all applicable sections which contain his key. This can become quite cumbersome and requires a facility for extremely fast scanning and comparing of text to a chosen key in order to be completed in a reasonable amount of time. HyperCard provides such a facility which, while not comparable to dedicated text scanning machines or mainframe computers, never the less produces satisfactory results for a personal (micro) computing environment.

The goal of TIM and indeed the goal of the ARGOS project is to demonstrate the feasibility of using "off the shelf" technology to produce a product that can be used on board ships today. Determining how this software will perform in a shipboard environment and defining the Military Specifications for the hardware remains to be done. However, preliminary development of ARGOS has shown that many of the short comings of current systems like SNAP II such as the lack of a graphical interface, can be solved without the need for specialized hardware/software. Also, by providing an easier to use interface ARGOS will

significantly reduce the amount of time required for training that plagues systems like SNAP II.

II. THE PROBLEM STATEMENT

A. BACKGROUND

Modern war ships in the American Navy are some of the most complicated pieces of machinery built in this century. Many major systems on board, notably those involving weapons and engineering, are computer controlled and require large volumes of technical information for proper operation and maintenance. More often than not the task of gathering the information required to perform even simple corrective or preventative maintenance takes more time than the job itself, which can lead to several problems.

Those sailors who are conscientious enough to perform the maintenance properly often find themselves with less hours in a day then they need to get their work done. After all, maintenance is only one part of a sailor's job. Personal and professional development which are essential to retaining quality people in the service may suffer as a result of long unproductive work hours. At the very least job satisfaction will begin to decline.

Some sailors may decide that the time required to research the facts pertaining to a certain job is not worth the effort and may perform improper or incomplete maintenance, despite the fact that doing so is punishable under the Uniformed Code of Military Justice. This is a serious problem because not only one's career is affected, many lives which depend on properly operating equipment may be affected if that equipment fails due to improper maintenance.

A third problem that could result is lost or stolen technical information which is one of the easiest problems to solve in a computer based information management

system. If a crew member performs a particular maintenance task on a daily or weekly basis, he may decide to keep all or part of the required technical documentation in his work space or personal locker. He may "hoard" technical manuals. This means that if someone else needs that information it will be temporarily or permanently unavailable. Since a computer based system would be available to all hands and hard copies of desired manual pages could easily be generated, a shipboard technical information management system could be a valuable tool.

B. PROBLEMS TIM SHOULD ADDRESS

1. Quantity of Information Provided

When designing a text retrieval system there are two pitfalls to be aware of: providing too much information, and not providing enough information. When too much information is provided the system quickly looses effectiveness and it's almost easier to use a hard copy book rather than a computer based system. The reason for using the computer in the first place is to narrow down choices to only those which are relevant to a particular query. So providing too much information is a waste of the user's time.

Designing a system that does not provide enough information presents a different problem. Many people find computers difficult to use and understand, for them computers are mysterious "black boxes." If a query is made about a particular subject these users feel that the answer provided by the computer must be exhaustive and represent all available information. If the computer has missed an area of a subject that the user wants to see, then this system provides too little information. TIM should be designed to avoid the information quantity problem, which means striking a balance between over informing and under informing. When computer

based information management systems are initially placed on ships, the tendency may be to keep hard copy technical manuals on hand to periodically "check" the computer for accuracy or completeness. This defeats TIM's purpose if it is dependent on the very manuals it is designed to replace.

2. Type of Information Provided.

What exactly should TIM tell a user about a particular subject? For example: given that the technical manual for the ship's Gas Turbine Engines is chosen, how does TIM go about flagging all information dealing with the *Transfer Gearbox*? As stated in the introduction, HyperCard has an extremely fast text searching algorithm, so with this TIM should be able to provide some way of presenting "useful" information about a text search on the string *Transfer Gearbox*. Some technical manuals arrange subjects by: chapters, sections, subsections, pages and paragraphs, with the paragraphs being numbered. So useful information in this case could be the page and paragraph number of every occurrence of *Transfer Gearbox*. However TIM should also be able to provide some context in which the information was found. To simply say that *Transfer Gearbox* is located on page 22, paragraph 3 is not as meaningful as saying: "page 22, paragraph 3, under the section on External Engine Components." TIM should provide the user with enough amplifying information to make the results of his search meaningful.

Since the system will be searching text to find occurrences of a particular string key, TIM should provide four modes of operation in order to enable the user to narrow the focus of his search . They are:

- Whole Key - which means choose only words which match the key letter for letter.
- Contains Key - such as "Compressor" and "Prediction" contain "pre".

- Begins with Key - choosing words which begin with the selected key, such as "Turbine" begins with "Tur".
- Ends with Key - such as "compaction" ends with "tion".

3. Converting Printed Material

Some of the most commonly used technical references on a ship are the Naval Ships Technical Manual (NSTM) series. These books are published under the direction of the Naval Sea Systems Command and cover a wide variety of equipment and subjects. One of these manuals would make a good test model for TIM, yet how should the electronic book mimic the printed one? Once the format has been determined how should it be entered into the computer? Ideally, in these days of modernization all publishing tasks should be handled by computer and the information would be easily transferrable from one electronic form to another through some sort of conversion. Many newer manuals for modern equipment are produced in such a fashion, yet a large volume of technical information that was printed many years ago for older equipment exist only in paper form. Without having an operator sit in front of a computer and actually type a 300 page manual over again, TIM should provide a way to convert printed material into an electronic format.

There is also the issue of pictures. Not only pictures, but graphs, diagrams, artwork, etc. All must be made available to the user just as they are in the printed manuals. This is one of the biggest drawbacks of current text oriented systems like SNAP II and should be a focal point of TIM. Since HyperCard allows the developer to produce a graphical interface for his applications, and since graphics are an integral part of ARGOS [Ref 1], TIM should handle graphics in a smooth and natural way.

4. User Interface Issues.

The user interface is one of the most important, yet one of the most overlooked areas of any computer application. Since TIM is an extension of ARGOS much of the look and feel of the system has already been determined. However there are some areas of a text retrieval system, with a graphical interface, that need to be addressed. Examples of these issues include:

- How will the user enter his query?
- How will user select a particular manual/library to search?
- Does the system take steps to prevent the user from making mistakes?
- When mistakes occur, how easily can the user recover from them?
- When TIM has returned requested information, how will the user select from the available choices?
- Since it has been stated that TIM should provide four modes of operation, the selection of the mode should be easy to change for each query, how will this be handled?

These issues are important because even the most efficient and full featured program will gain little acceptance if it is hard to use.

C. CONCLUSION

Although TIM is only one of many ways to design a text retrieval system, it is the best way to design a system that will integrate in a seamless fashion with ARGOS while still meeting the desired functional requirements. Initially it appears that the concept of a Technical Information Manager would be a straightforward problem with a very difficult solution. The problem would be: "given a book, how do you represent it in an electronic format and retrieve useful information about a particular subject?" The solution would then involve the application of classic Computer Science techniques. As it turns out, the difficulty of TIM's development

is to define the problem more precisely. This is because people tend to have varying opinions about what constitutes "useful" information when discussing text retrieval systems.

III. DEVELOPMENT ENVIRONMENT (HYPERCARD)

A. WHY HYPERCARD WAS CHOSEN TO DEVELOP TIM.

The primary reason TIM was developed in HyperCard is because it was the development environment of the pilot version of ARGOS. This permitted the reuse of some previous constructs, and resulted in a program that maintained the same look and feel as the pilot. However, there are two additional reasons why HyperCard was a desirable choice, the first of these being its modularity. Functional areas can be built totally independent of ARGOS and integrated with a minimal amount of modification to it (ARGOS). This means that in order to develop a module the programmer need not have an intimate understanding of ARGOS' internal structure, only a good familiarity with its functionality.

The second reason for choosing HyperCard is its rapid prototyping characteristic. The amount of time required to produce working code is significantly less than that of a conventional programming environment like C, because HyperCard automates the development of the graphical interface. All windows and other tools which a programmer would incorporate into his application are arranged in a menu. The task of producing the code for these objects is handled by the HyperCard environment in cooperation with the Macintosh operating system. The developer simply selects an object and places it in his program where desired. This feature often gives HyperCard the misnomer of an "Object Oriented Programming (OOP) language," because although HyperCard does possess some powerful capabilities, it is not a true OOP language.

B. FACILITIES PROVIDED BY HYPERCARD

1. Stacks, Backgrounds, Cards, Fields and Buttons

A stack in HyperCard is analogous to a main program definition in a language like C. The creation of an application begins with a new stack and all of the user defined code, icons and external resources are contained in the stack. Figure 1. is an information dialog box about a stack, showing a typical layout. An

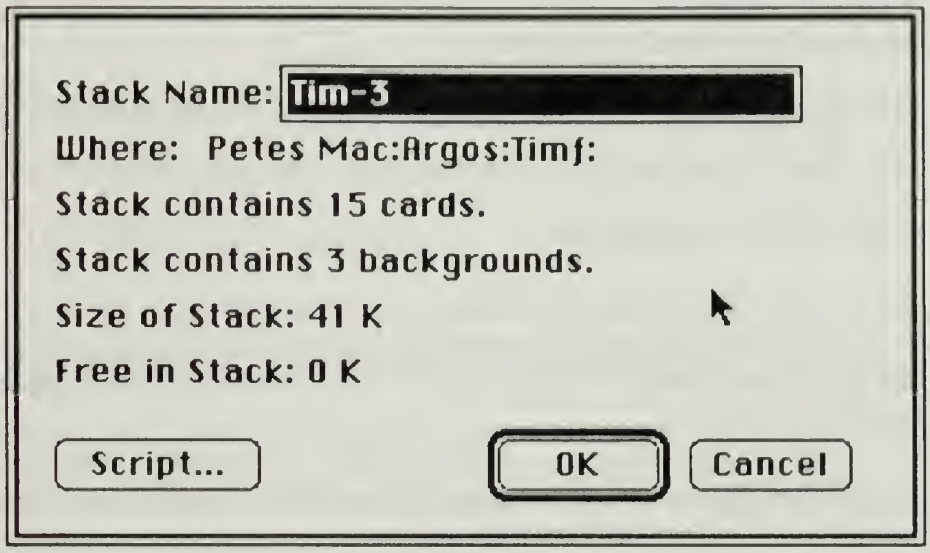


Figure 1. Typical Stack Layout.

advantage of HyperCard is that a special stack, called the *Home Stack*, is designated as a central location for commonly used resources. When a new stack is created, it will default to the Home stack when looking for a particular icon or other external resource. By customizing the Home stack, those resources which are common to the various modules of ARGOS can be centrally located. This not only prevents the duplication of shared code, it results in an easier to maintain software system.

Backgrounds are a useful and powerful construct in HyperCard. When a stack is created, cards of a particular group may be identical except for the data contained in their fields. This group of cards is said to be *homogeneous,* because all share one set of attributes (number and type of fields, buttons, etc). Rather than recreating these attributes for each card, the cards draw on a set of common attributes called a background with all cards of a particular background being based on the same structure. When a new card is created, a template will be generated which can then be customized as desired.

The remaining objects form the basic building blocks of all stacks. They are simple to create, yet are very powerful programming tools which act as the user interface to HyperCard's capabilities:

- Cards: A template upon which buttons, fields and artwork are arranged. The analogy being very similar to a series of 3" X 5" index cards (a group of which is also called a stack.)
- Fields: Used to enter, display and modify text. The attributes of the field, as well as all other HyperCard objects, can be modified either from a menu during stack creation or dynamically during runtime.
- Buttons: The control mechanism which allows a user to initiate various actions in HyperCard. They are graphical representations (icons) which are activated by positioning the cursor over them and pressing the mouse button.

2. Graphic Design Tools

HyperCard provides the basic tools necessary to design and manipulate graphics. While not as powerful as dedicated graphics programs like SuperPaint™ or MacDraw™, these tools make it possible to create basic designs and manipulate graphics created with other programs. Figure 2 shows HyperCard's various graphic tool menus.

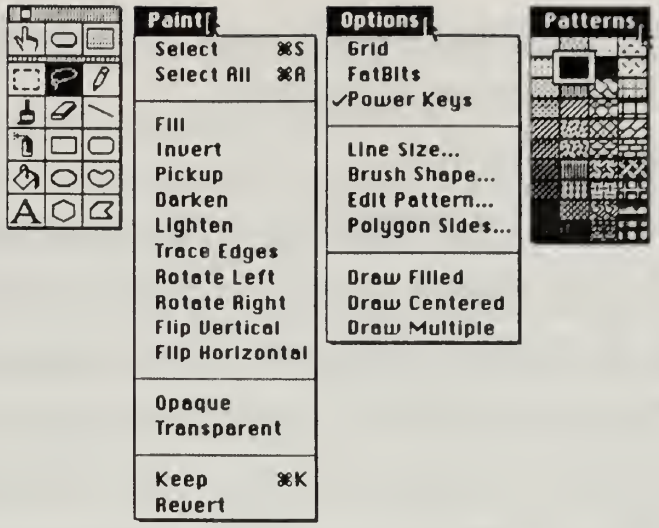


Figure 2 HyperCard's Graphic Tool Menus

3. HyperTalk and Scripting

A *script* in HyperCard is the source code that a programmer writes. The language is known as HyperTalk, and all five objects (stacks, backgrounds, cards, fields and buttons) can have scripts. When an object is created the programmer uses a menu command to get information about its various characteristics. A dialog window comes up which includes a button called *Script...* (refer back to Figure 1.) Clicking on this button brings forward a window into which a HyperTalk script for that object can be entered.

An important advantage that HyperTalk has over other programming languages is that it tries to model plain English as closely as possible. The syntax of the language is very easy to read and as in English it can be written in different ways, while still having the same meaning. For example: a line of HyperTalk code to put the first word of a variable called *sentence*, which contains the string "The Quick Brown Fox" into a variable called *myvar* could look like this: *put word one of sentence into myvar.* The command could also be written as: *put the first word of*

sentence into myvar. to produce the same result. For a thorough description of HyperTalk, as well as HyperCard, refer to [Ref 2].

4. XCMDs and XFCNs

While HyperTalk is powerful, there are some programming tasks it does not perform well. For example it does not provide useful functions that communicate with the Macintosh's serial ports. Other languages, like C, have the ability to link portions of code written in assembly language to the main program. HyperCard provides the same type of functionality with XCMDs (External Commands) and XFCNs (External Functions.) They are code resources written in a language like C or Pascal, and installed into a stack along with other resources such as icons. Once installed they can be called from HyperTalk just like any other function. This means that ARGOS is not limited to the internal HyperCard environment, and thus, can be extended easily.

C. HYPERCARD COMPARED TO OBJECT ORIENTED PROGRAMMING (OOP) LANGUAGES.

1. Object Oriented Programming (OOP) languages.

An Object Oriented Programming language is different from a procedural language like C or Pascal. In a procedural language, functions are created which operate on certain types of data with the results being passed back to the calling program. In an OOP language the data and functions are located in the same "object." To operate on the data a *message* is sent to the object instructing it to invoke one of its functions. The only way to access the desired data is to use one of the functions contained in the object or another function especially declared to have access to the data. This leads to increased modularity as well as increased data

security. Objects are one of five basic building blocks of OOP languages, the others being: classes, inheritance, messages and methods.

Simply put, a class in an OOP language is a collection of all objects of the same type. Objects are said to be *instances* of classes, which means that each instance (object) of a class will share one common set of functions. The advantage is that each time an object is created it automatically has at its disposal all of the functions needed to operate on its data. Also, when objects are created they will behave in a predictable manner which is determined by the class definition. This prevents accidental use of functions written for an object of one type (called member functions) on objects of another type. For example: A member function defined for a class called *rational numbers* cannot operate on objects of a class called *complex numbers*. However, one class can be used as an object in another class. This brings up the idea of inheritance.

Inheritance is a important aspect of all true OOP languages. The idea being that as classes are created, some classes may be defined as parts of others. Figure 3 is an example of inheritance.

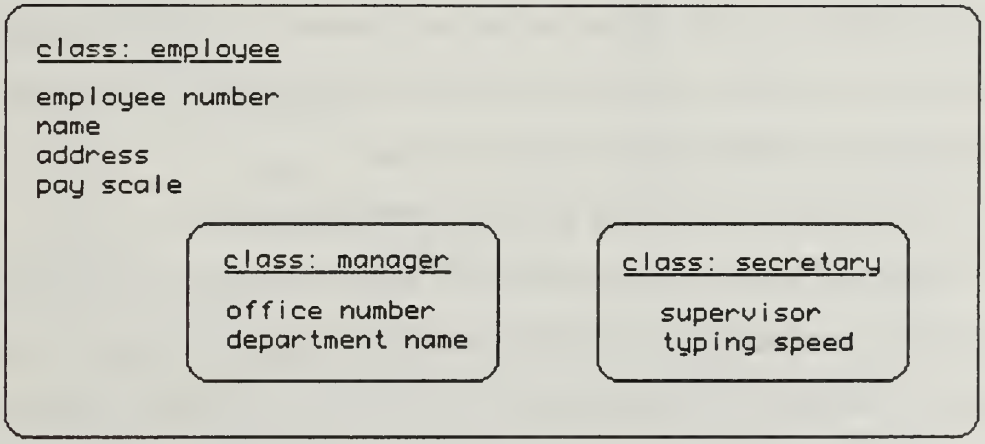


Figure 3. Inheritance

All objects of the class *employee* have: employee number, name, address and payscale attributes. Objects of the class *manager* have all the attributes that are defined in *employee*, plus: office number and department name. Objects of the class *secretary* also have: supervisor and typing speed. Inheritance is important because objects of sub classes, like *manager*, can now use member functions defined in the super class (*employee.*) For example: if the class *employee* defines a member function called "print_name_and_address", this function can be invoked by objects of the class manager. In this way the *manager* class does not have to define a separate function to print out the name and address of one of its objects.

Methods and messages are the two remaining, and easiest to explain, concepts in OOP languages. A method is a function, either public or private, defined as part of the class definition. Public methods can be invoked from anywhere in a program that the class definition is visible, and private methods can only be invoked from public methods or from other private methods. In C++, private methods can also be invoked outside of the class definition by functions which are granted special access to a class' private data. These functions are known as *friend* functions. In both cases methods are invoked by using messages. A message is sent to an object to command a method to initiate some action on the data, with messages and methods being related one-to-one.

2. Why HyperCard is not a True OOP Language.

HyperCard has some parallels with OOP languages, yet is different in many respects compared to its OOP counterpart. The best way to contrast HyperCard to traditional OOP languages is to view how HyperCard handles the five

basic areas of OOP programming discussed earlier: objects, classes, inheritance, messages and methods.

Objects in HyperCard follow the same idea is objects in OOP. An object contains the methods and data required to carry out its function and is invoked with messages. In this way HyperCard is very similar to OOP. HyperCard is different in that it offers only five objects, which are defined by the system: stacks, backgrounds, cards, fields and buttons. There are no user-defined objects in HyperCard which would be considered a serious limitation in true OOP languages.

HyperCard has no strong parallel to the concept of OOP's classes. There is no class of *fields*, for example, in which every field is an instance of the class. Even though all fields may contain similar attributes, there is no scheme in which they all belong to a certain class and share a set of attributes and data. Rather HyperCard is arranged as a scattering of the five system defined objects. This becomes important when discussing inheritance in HyperCard.

There is no strict inheritance in HyperCard as there is in OOP, but there is a hierarchy which messages follow as depicted in Figure 4. (taken from [Ref 2])

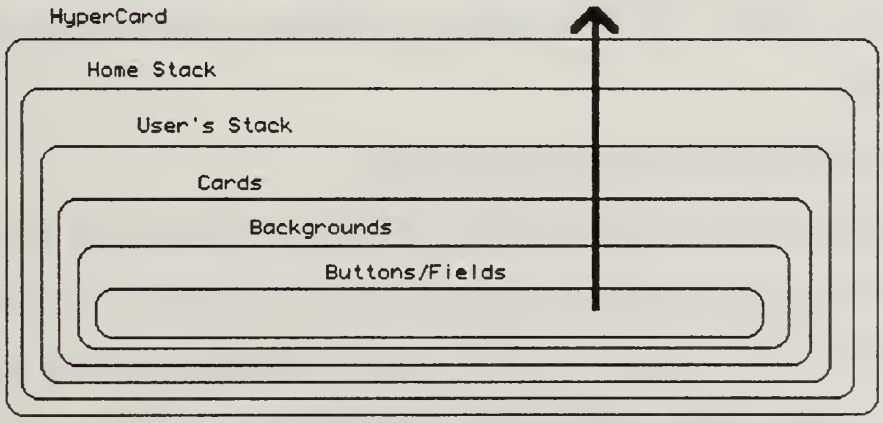


Figure 4. Hierarchy of Message Travel in HyperCard.

When a message is originated it travels up the hierarchy from its current level until a method with the same name is encountered. If none is found, the message exits the system. This is a hierarchy, which is different from inheritance where sub classes can use the methods of their super class. Just because a field in a HyperCard stack is designed to handle a particular message does not mean that all fields in that stack can handle the same message.

Messages in HyperCard are the same, conceptually, as messages in an OOP language. They are passed between objects and direct an action to take place through the use of methods. Methods in HyperCard come in one of two forms: functions and handlers. A function is a section of code written to take zero or more parameters as input and return exactly one value. A handler is a section of code written beginning with the key word "on" followed by a handler name, for example: "on mouseUp." In this case, depending on where the code is placed in the hierarchy, it will react to a *mouseUp* message sent from an object. A characteristic of HyperCard is that methods of one object can manipulate other objects. For example: a method written into a button can write text into a field, without using one of the field's methods. This is an important difference between HyperCard and OOP languages.

D. ADVANCEMENTS IN HYPERCARD.

HyperCard version 1.2.2 was used to develop TIM and while it was quite powerful it should be noted that HyperTalk is an interpreted, not a compiled, language. Each time a stack is used the HyperTalk scripts must be parsed for correctness. This tends to make some of HyperCard's features run slowly on Macintosh Plus and SE computers, and somewhat better on Mac II, IIX, IICX and

IICi computers. This also means that for a stack to run, a copy of HyperCard must be present. These limitations, while somewhat unwelcome, are acceptable for developing ARGOS and TIM as prototypes and should be addressed in future releases of HyperCard as well as developers of future versions of ARGOS.

IV. DESIGN AND IMPLEMENTATION

A. TECHNICAL MANUAL CONVERSION

1. From Printed to Electronic Form

As discussed in the problem statement, TIM should provide a way to convert printed material into an electronic form, without having to do a lot of typing. This function is not provided by TIM, but rather by an Optical Character Recognition (OCR) program called Omni Page[1] and a Hewlett Packard ScanJet[2] scanner. Together they take a printed page of text and convert it into an ASCII text file.

First the desired page is placed in the scanner and a high (300 dpi) resolution image of it is made, which is stored in main memory. Omni Page then quickly converts the image into ASCII text which can be formatted and saved as desired. Omni Page was crucial to TIM's development, and allowed for the conversion of the entire text of Volume One of the Propulsion Plant Manual for FFG-7 class ships in less than 80 hours. A complete description of the scanning process can be found in the Omni Page user's guide.

2. Index Stack

While HyperCard's text scanning and searching routine is adequate, it is plagued by the same problem that affects mainframe computers: it starts to lose effectiveness if the amount of text being considered is very large. An entire manual

---

[1]Omni Page is a registered trademark of Caere Corporation.

[2]ScanJet is a registered trademark of the Hewlett Packard Corporation

can be searched from start to finish in a reasonable amount of time, but when designing TIM it seemed there should be some way to narrow the focus of the search to improve system performance. This lead to the development of the *Index Stack.*

As discussed earlier, the smallest division of a Naval Ships Technical Manual (NSTM) is the paragraph. In the beginning of the manual there is an extensive table of contents which lists page and paragraph number for each subject contained within. It is a detailed representation of the contents of the book and allows TIM to forego the need to do a page by page search of the manual. The table of contents is simply arranged in a separate stack and when information about the manual is requested, this is where the search takes place. There are a total of 30 printed pages in the table of contents of the technical manual that was modeled, which is considerably less than the 300 or so pages in the manual itself. By searching the index stack instead of the manual there is about a 35% reduction in search time.

Figure 5 is an example of how entries in the Index Stack are arranged. Characters 1 thru 4 of each line are part of a coding scheme which allows TIM to process the results of a search more quickly. The first character identifies each line as one of four types:

- + Chapter Heading
- * Subject Heading
- # Topic Heading
- > Topic

The remaining characters (2-4) are the line number which corresponds to the super heading for a given line. For example (referring to Figure 5) the line:

#4 1-6 Gas Turbine Modules (GTM's) 1-1.

is a topic heading located in paragraph 1-6 of page 1-1. Its super heading is located on line 4, which is:

**1 1-4 PROPULSION SYSTEM 1-1.*

This is a subject whose super heading is the chapter name located on line number one. Since chapter names are the highest in the pecking order, they have line zero as a super heading. A more through description of TIM's search processing algorithm is provided in section B.1 of this chapter.

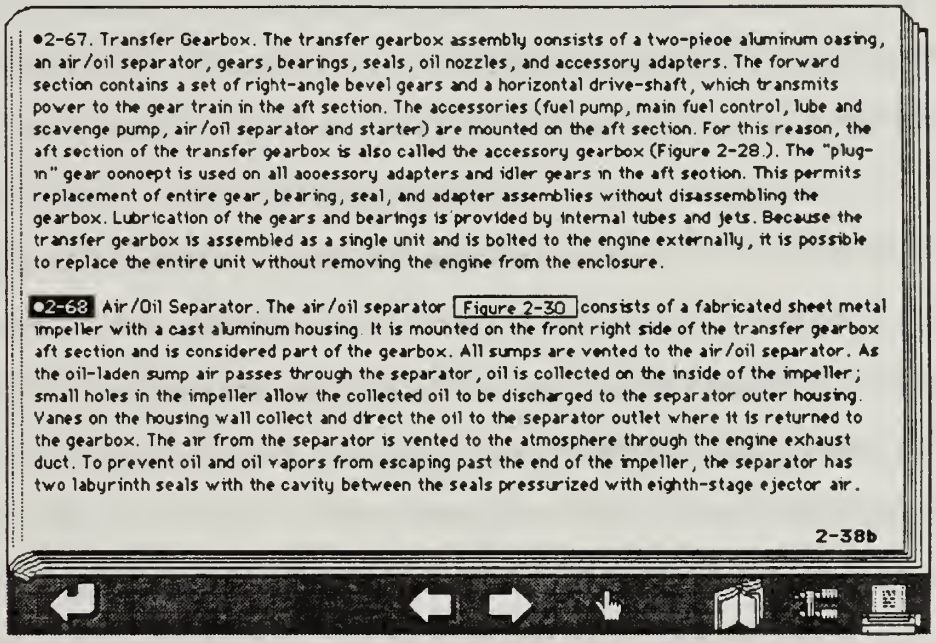
```
+0    1. OVERALL DESCRIPTION OF FFG-7 CLASS ENGINEERING PLANT 1-1

*1    1-1       INTRODUCTION 1-1
*1    1-4       PROPULSION SYSTEM. 1-1
*4    1-5       GENERAL 1-1
*4    1-6       GAS TURBINE MODULES (GTM's) 1-1
*4    1-8       PROPULSION GAS TURBINES ENGINES 1-1
*4    1-9       MAIN PROPULSION INTAKE AND EXHAUST SYSTEMS 1-4
*4    1-10      MAIN REDUCTION GERR (MRG) 1-4
*4    1-11.     CONTROLLABLE PITCH (CP) PROPELLER 1-4
*4    1-12.     LUBE OIL SYSTEM 1-4
*4    1-13.     FUEL OIL SYSTEMS 1-4
*4    1-17.     BLEED AIR SYSTEM (BAS) 1-5
>13   1-18.     Anti-Icing 1-5
>13   1-19.     Prairie Air 1-5
>13   1-20.     Masker Air 1-5
>13   1-21.     Emergency Cross-Bleed Gas Turbine Starting 1-5
*4    1-22.     AUXILIARY PROPULSION SYSTEM (APS) 1-5
*1    1-23.     ELECTRICAL SYSTEM 1-6
```

Figure 5. Index Stack Arrangement

3. Text Stack

The remainder of the technical manual is placed in the *Text Stack*, and its pages are formatted as shown in Figure 6. The layout of the stack is patterned after the printed manual, with a "book like" graphical representation of the pages. Since one printed page will not fit into a card in the text stack, pages are split up with the

appropriate lettering suffix (a,b,c,...) added to the page number on the bottom right hand corner. (A scrolling field could have been used to allow a one-to-one mapping of pages from the printed manual to the stack, but scrolling fields are less intuitive than a "one thought per page" metaphor.) In addition to displaying the text of a manual, the Text Stack also provides three useful tools which make finding information easier, they are: reference to figure linking, auto jump to any page and string search.



•2-67. Transfer Gearbox. The transfer gearbox assembly oonsists of a two-pieoe aluminum oasing, an air/oil separator, gears, bearings, seals, oil nozzles, and accessory adapters. The forward section contains a set of right-angle bevel gears and a horizontal drive-shaft, which transmits power to the gear train in the aft section. The accessories (fuel pump, main fuel control, lube and scavenge pump, air/oil separator and starter) are mounted on the aft section. For this reason, the aft section of the transfer gearbox is also called the accessory gearbox (Figure 2-28.). The "plug-in" gear oonoept is used on all aooessory adapters and idler gears in the aft seotion. This permits replacement of entire gear, bearing, seal, and adapter assemblies without disassembling the gearbox. Lubrication of the gears and bearings is provided by internal tubes and jets. Because the transfer gearbox is assembled as a single unit and is bolted to the engine externally, it is possible to replace the entire unit without removing the engine from the enclosure.

•2-68 Air/Oil Separator. The air/oil separator Figure 2-30 consists of a fabricated sheet metal impeller with a cast aluminum housing. It is mounted on the front right side of the transfer gearbox aft section and is considered part of the gearbox. All sumps are vented to the air/oil separator. As the oil-laden sump air passes through the separator, oil is collected on the inside of the impeller; small holes in the impeller allow the collected oil to be discharged to the separator outer housing. Vanes on the housing wall collect and direct the oil to the separator outlet where it is returned to the gearbox. The air from the separator is vented to the atmosphere through the engine exhaust duct. To prevent oil and oil vapors from escaping past the end of the impeller, the separator has two labyrinth seals with the cavity between the seals pressurized with eighth-stage ejector air.

2-38b

Figure 6. Text Page Layout

Reference to figure linking is best described as a link between the reference to a figure and the actual figure itself. For example: a user is reading the technical manual page shown in Figure 6 and comes to the first sentence of paragraph 2-68 which reads:

Air/Oil Separator. The air/oil separator Figure 2-30 consists of a fabricated sheet metal impeller with a cast aluminum housing.

However Figure 2-30 may be several pages away either forward or backward and getting to it could involve a great deal of electronic "page turning." To solve this problem TIM provides a facility for instantly jumping to figures and back again.

The characters; *Figure 2-30* , have a box around them which is actually a button. This button contains a script which has the card identification number of Figure 2-30 in it. Clicking on this button causes the user's current screen to be replaced with Figure 2-30. Conversely, clicking anywhere on the figure causes the user to return to his original screen. The advantage of this over a conventional book is that the user need not keep his place when "flipping" to a figure. The system will keep track of where he was and return there at the appropriate time. This technique, used to display tables as well (i.e. *see table 4-5*), is accomplished with HyperCard's facility for automatically linking buttons to cards, as described in [Ref 2].

<u>Auto jump to any page</u> is analogous to an electronic page turner. Clicking on the open book icon (bottom of Figure 6) brings forward a dialogue box which asks the user what page he would like to go to. After the page number is entered, the system uses an extremely fast binary search algorithm to locate the desired page. If the page entered is not present in a given manual, the system will locate the highest numbered page which is less than the target page. The binary search can be used because, like the pages of the manual, the cards of the Text Stack are arranged in sequential order.

<u>String Search</u> is a simple yet powerful feature. Once the user has entered the technical manual, he may wish to search for a particular word. Clicking on the

binoculars shown in Figure 6 allows the user to enter and search for a given string. The system will search the text of the manual and place a box around the first successful match. Subsequent matches can be found by pressing the return key. This is slower than performing a search of a manual's index, but can be useful to locate hard to find things like: part names or numbers. Defining a search by using the Index Stack is described in section B.2 of this chapter.

Reference to figure linking, auto jump to any page and string search are strictly part of the Text Stack, where their HyperTalk code is placed, and are separate from TIM itself. However, since these features will be available to every manual in TIM's library, future versions could relocate this code to a central location to avoid duplication as new manuals are added.

B. SEARCHING FOR INFORMATION

1. The Search Algorithm

A powerful tool that HyperCard provides is the ability to simulate keyboard input using the *type* command. For example: issuing the command *type return* from within a script causes a carriage return to be sent to HyperCard. Any legal command which is in HyperCard's designated command line window, called the Message Box, will then be executed just as if the user hit the return key. Placing desired commands in the Message Box and sending carriage returns from within a script allows for easy execution of any HyperCard command. This feature is central to TIM's search algorithm.

When a search is initiated, TIM takes the technical manual title field (located at the top of the customized table of contents card) and appends the string ".idx" to it. This indicates that the search will take place in the Index Stack of the chosen manual. (Index Stacks have the same name as the Text Stacks, except with

the ".idx" extension) TIM then makes the screen coordinates of the Message Box equal to (-100,-100), which ensures that as its contents are manipulated, the user will not be able to see it. The string "Find ", followed by the user's desired target string, is placed in the Message Box. (*Find* is a HyperTalk command that initiates an ASCII search for a particular string.) TIM then goes to the Index Stack of the chosen manual and sends a carriage return to HyperCard, executing the *Find* command in the message box. HyperCard searches the Index Stack for the first occurrence of the chosen string. If none is found, then a system defined variable called *TheResult* will contain information about why the search failed, and TIM returns to the main window to inform the user.

If *TheResult* is empty, an occurrence of the user's string was found. TIM then saves the value of another system variable called *The FoundChunk,* which indicates: card, field and beginning to ending character number of the found string. A carriage return is sent to HyperCard to continue the search, and as new occurrences of the string are found, the updated value of *The FoundChunk* is compared with the value first encountered. If they are equal, then the first occurrence of the string has been found again and the search is terminated. If not, then *type return* commands are repeatedly executed until the first value of *The FoundChunk* and the current one are equal.

As matches to the user's string are found, the line number where they are located is automatically stored in a system variable called *The FoundLine.* Using this variable, characters 2-4 of the target line can be examined to determine what line the appropriate super heading resides on. (recall that char 1-4 of each line of the Index Stack is part of the coding scheme described in section A.2.) This super heading line number is saved, and a copy of the target line is passed to a formatting

routine which pads it with blanks based on its position in the hierarchy. The desired result is to have lines cascading to the right as follows:

```
+ Chapter Headings
        * Subject Headings
                # Topic Headings
                        > Topics.
                # Topic Headings
                        > Topics
                        > Topics
```

The formatting routine strips characters 1-4 of the input line, pads it with the appropriate number of blanks, and returns the result which is placed in a variable called *SearchResults*.

If the line number of the super heading is non-zero, then the super heading is processed in the formatting routine and placed "before" *SearchResults* (Note: in HyperCard, placing "B" *before* a variable holding "AT" results in "BAT".) This is repeated until a super heading numbered zero is encountered, or a super heading is encountered that has already been processed. It can be assumed that if a topic heading of a particular topic has already been processed, then *all* the super headings for that topic have been processed as well. This pruning heuristic is quite reliable and significantly reduces search processing time.

When the string match and all its super headings have been processed, the next occurrence of the string is found, processed the same way, and placed "after" *SearchResults*. When all the searching is complete, *SearchResults* will contain the

customized table of contents for the user's query. This is passed back to the customized table of contents card of the TIM stack, and placed in the appropriate slot as described in section B.3.

2. Defining a Search

The opening window of TIM is shown in Figure 7. Getting information from a manual involves five steps:

- Defining the particular library of manuals to be searched.
- Choosing a manual from that library
- Defining the search key
- Defining the search type
- Making a selection from the results of the search.

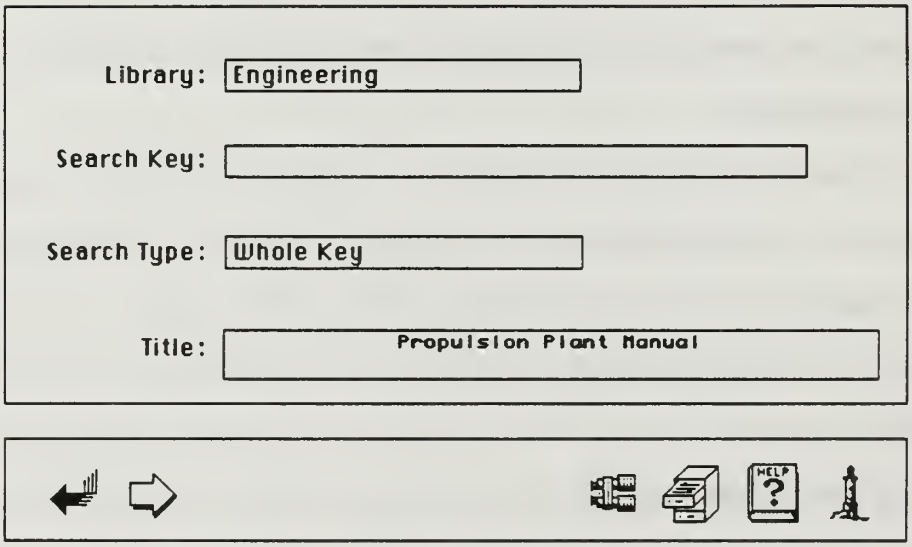


Figure 7. Opening TIM Window

The idea of a library in TIM involves arranging shipboard technical manuals according to the five basic departments: Weapons, Operations, Navigation/Admin, Engineering and Supply. When the user begins a session with

TIM he first makes a selection from the list of available libraries by clicking and holding down the mouse button while the cursor is positioned over the field for library selection. This will bring forward a "Pop-Up" menu from which the user makes a choice by dragging to the desired selection (with the mouse button still down) until it is highlighted and releasing the button as displayed in Figure 8.

When the user has selected a library the system will customize the title menu to display only those manuals which are part of the chosen library. The user

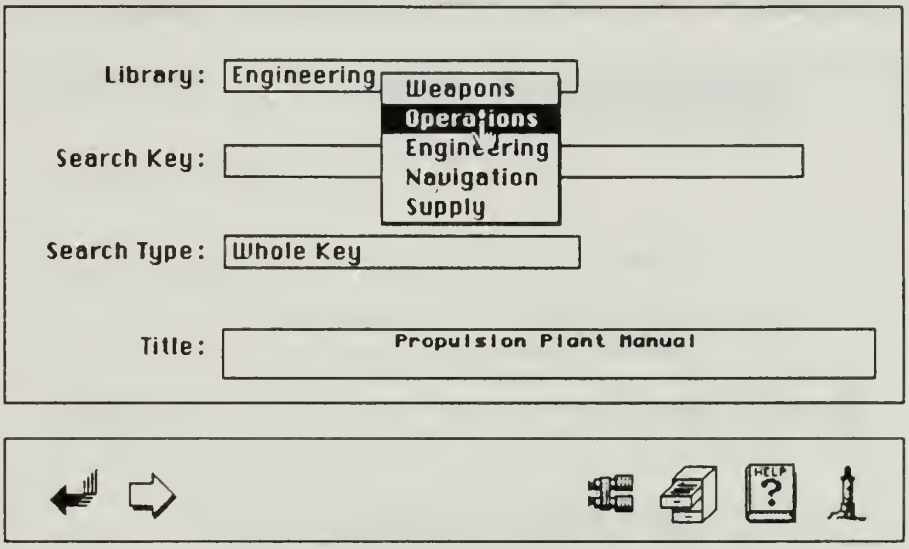


Figure 8. Library Selection

then selects a technical manual in a similar fashion to selecting a library, this time positioning the cursor over the technical manual title field (Figure 9).

Defining a search key and search type are accomplished the same as library and manual selection. In defining a key the user can create a new one, or open a file which contains the customized table of contents cards of a previous TIM session. In this way he can quickly recall information that is required on a regular basis without

having to actually do the search all over again. The search type is chosen from a list of four:

- Whole Key, which means choose only words which match letter for letter
- Contains Key, such as "Compressor" and "Prediction" contain "pre",
- Begins with Key, choosing words which begin with the selected key, as "Turbine" begins with "Tur"
- Ends with Key such as "compaction" ends with "tion".

Key and Type selection are show in Figures. 10 and 11. Note, key selection and matching are not case sensitive in TIM. (That is: "OPEN" is the same as "open" or "OpEn")

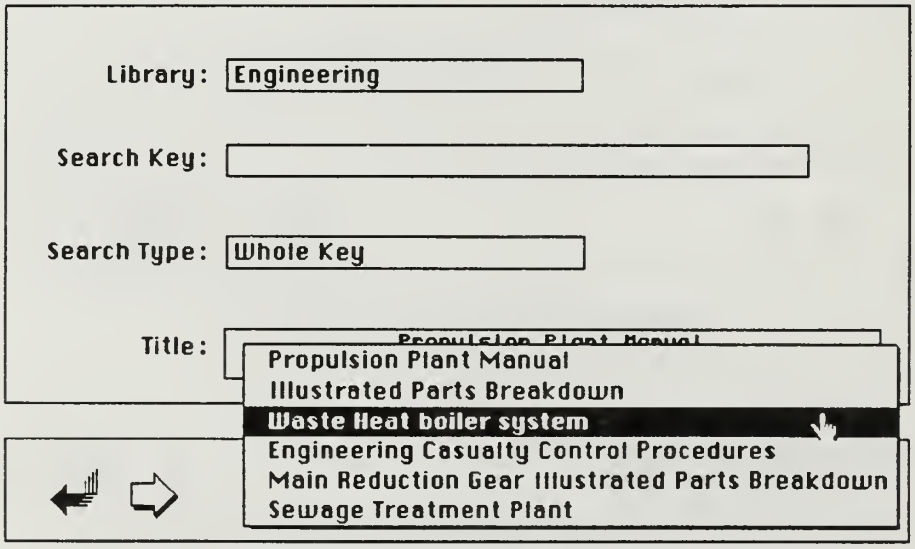


Figure 9. Technical Manual Title Selection.

Now that the library, manual, key and search type have been selected the search must be conducted. This is accomplished by clicking on the binoculars

located at the bottom of the opening screen (Figure 10.) The system prompts the user with a standby message and then accesses the appropriate manual, generating a customized table of contents. This customized table is a subset of the manual's real table of contents including only those entries which match the user's search criteria. Figure 12 is an example of a customized table of contents card, along with a description of the various information it contains.

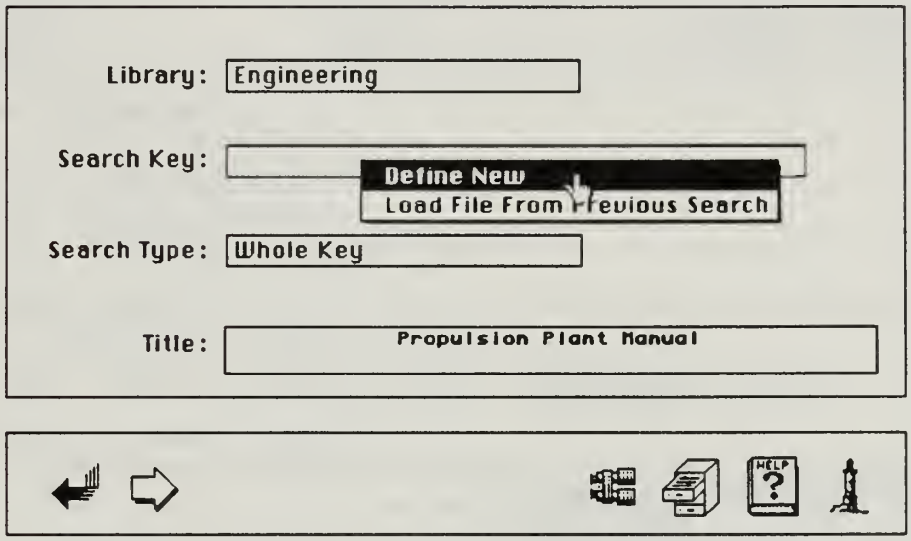


Figure 10. Search Key Selection

3. Choosing From the Results of a Search - The Customized Table of Contents.

To select a particular subject from a customized table of contents card, the user clicks on one of the lines in the card which will become hilited. By clicking on the open book icon with the magnifying glass over it (Figure 12), the user will go to the appropriate section (page and paragraph number) in the chosen manual. (Recall that technical manual pages are formatted as shown in Figure 6.)

The customized table of contents card has tabs located along the bottom which allow for the definition of up to nine different searches (Figure 12). For example the user can define a Whole Key search on the word "Oil" in the *Propulsion Plant Manual* of the Engineering Library, and a Begins With Key search on the word "Boat" in the *Small Boat Operating Guide* of the Operations Library. The first six letters of each search key will appear in a different tab at the bottom of the customized table of contents card. The user can then switch from one to the other by just clicking the tabs.

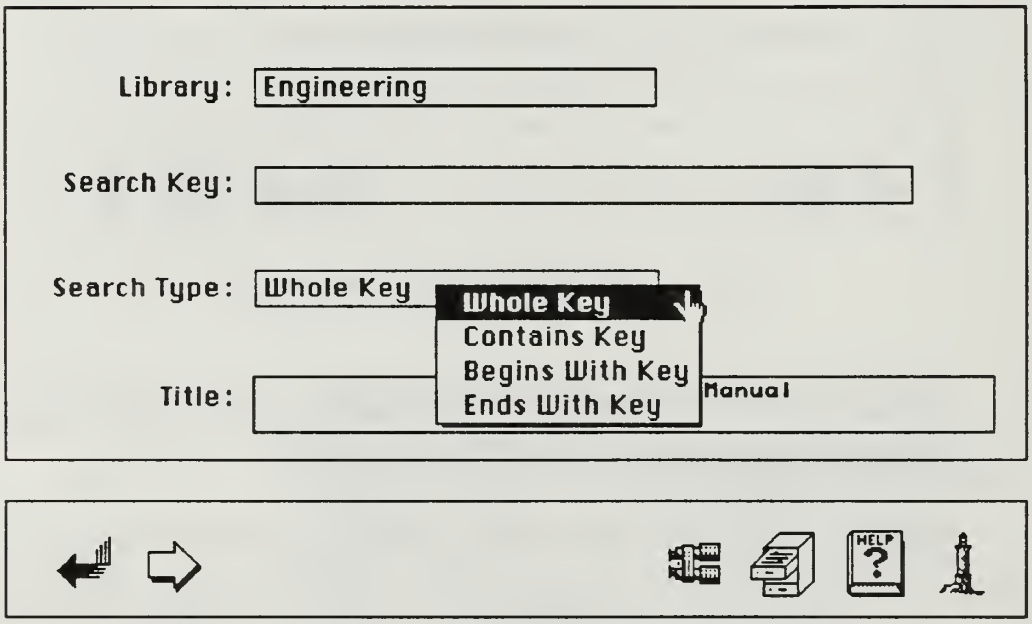


Figure 11. Search Type Selection

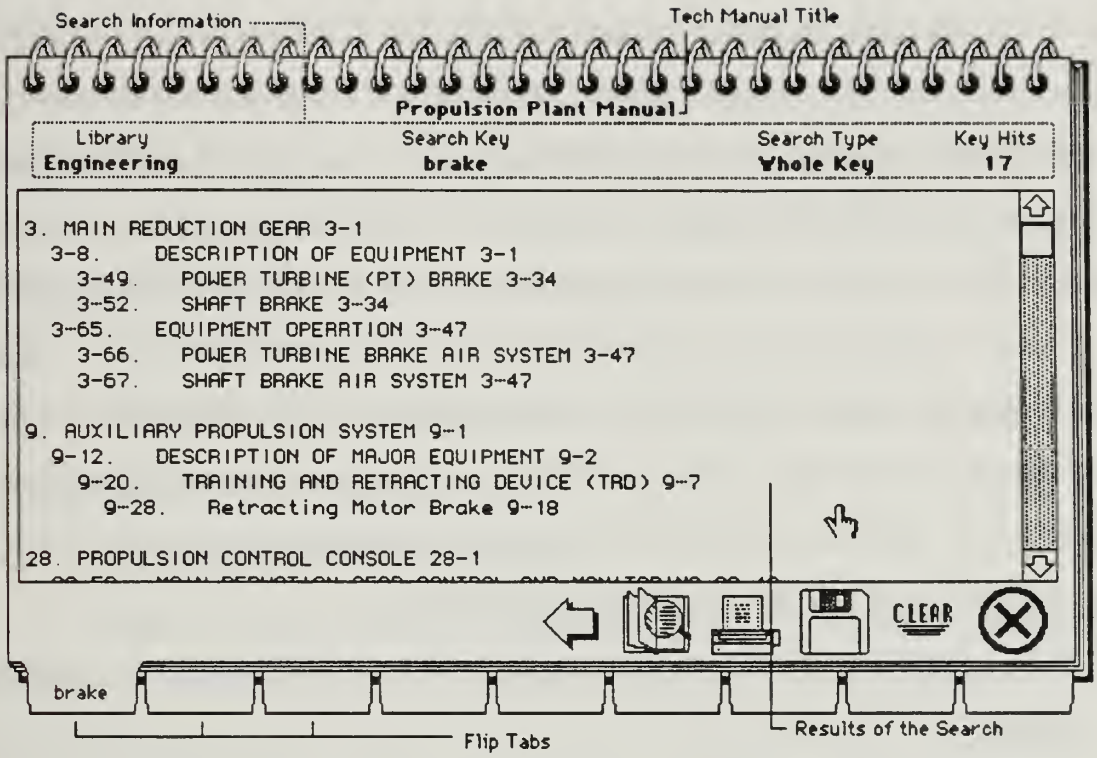


Figure 12. Customized Table of Contents card.

4. Managing the Customized Table of Contents

Since the customized table of contents contains 9 different cards, TIM needs some way to decide where the results of new searches should go. This is accomplished with the help of a hidden field, and some simple modulo nine arithmetic.

Each of the cards in the customized table of contents is named "Flip*x*", where *x* is a number between 1 and 9. When TIM is initialized, the number 1 is stored in a hidden field named *available* located in the background of TIM's Stack. This indicates to TIM that Flip1 is the location where a new search result is to be placed. When a search is conducted, and after getting at least one successful match to the query, TIM checks the value of field *available* to see that it is a 1. Upon transferring the results to card Flip1, TIM performs the following math on field available: *field available = ((field available) mod 9) + 1*. This ensures that the next search will be placed in card Flip2. If all nine cards become full, then the modulo nine arithmetic will start the count from the beginning.

5. Transition From the Customized Table of Contents to the Text Stack.

Making the transition from the customized table of contents to the text stack is a simple operation in TIM. HyperCard defines a *word* as a string of one of more characters with blanks, or new line characters, on either side. This allows a line to be addressed by words. Commands like: *put word 2 of myvariable into holder*, are legal and quite powerful in HyperCard.

When the user selects a topic, the line is hilited and saved in a hidden field. When the open book/magnifying glass icon is pressed, TIM strips the first and last words off the line in the hidden field. The first word is the paragraph number, and

the last word is the page number where the topic is located. TIM then takes the title of the technical manual from the top of the customized table of contents card, appends the string ".txt" to it, and opens the text stack. (recall Text Stacks and Index Stacks have the same name with either a ".idx" or ".txt" extension.) Once the stack is open, a message is sent to the binary search page finder to go to the desired page. When that has been accomplished, TIM executes the command: *find paragraphnum*, where paragraphnum is the first word of the selected topic. HyperCard will then search from the current card, forward, until the correct paragraph is found. Since one technical manual page takes up usually no more than 3 cards, the search will be quite fast. The user is now at the desired location.

6. The Function of the Various Buttons

The function of TIM's various buttons is described in Appendix A.

C. SYSTEM MAINTENANCE

1. Adding and Deleting Manuals

Shipboard personnel will be responsible for periodic addition and deletion of manuals. To add a manual to the system, its Index and Text stack must first be provided, and placed in the same folder as TIM. Secondly, its name is added to the system under the appropriate library. After that, it will show up when the user selects a technical manual title as described in section B.2. (recall that this menu will automatically configure to show manual titles based upon which library is selected.) Deletion of manuals simply involves removing their names from the library, and deleting their Index Stack and Text stacks.

Clicking on the filing cabinet icon in TIM's opening window (refer to Figure 11) brings forward the System Manager card shown in figure 13. New manual titles are simply typed into the "Titles" field, and TIM automatically makes

them available as input to the title selection menu of the opening window. Movement from one library to another is accomplished by using the directional arrows, and returning to the opening window of TIM is accomplished by clicking on the bent arrow in the bottom left corner.

2. Updates and Routine Maintenance.

Generally, shipboard personnel will not be required to perform any updates or changes to technical manuals. To maintain consistency fleet wide, it's better that only addition and deletion of manuals be allow at the shipboard level, except in urgent circumstances.

When a change to a manual does need to be made, a completely new manual can be issued by the appropriate authority. This can be sent to the ship via mail, on an optical disk, or if the change is urgent a ship can receive a new manual while in port via a telephone line and an inexpensive modem. This should handle a majority of the routine updates to technical manuals that may be required.

However, there may be a time when an urgent change must be made, that cannot wait until a ship reaches a port. In that case there should be a system manager onboard who has *Typing Access* to TIM. *Typing Access* is one of the five levels of user access to stacks defined in HyperCard:

- Browsing - which is a read only access.
- Typing - which permits a user to add/delete text.
- Painting - which permits a user to add/delete graphics.
- Authoring - which permits a user to modify the layout of a stack.
- Scripting - which permits a user to modify scripts.

If a someone has a certain access, then he also has all previous accesses as well. Most user's on the ship will not require more than *Browsing Access*. However, there should be at least one system manager with *Painting Access* (which would also include *Typing Access*) to modify manuals as required. Then, as soon as practicable, an updated manual should be obtained by the issuing authority to ensure fleet wide consistency.

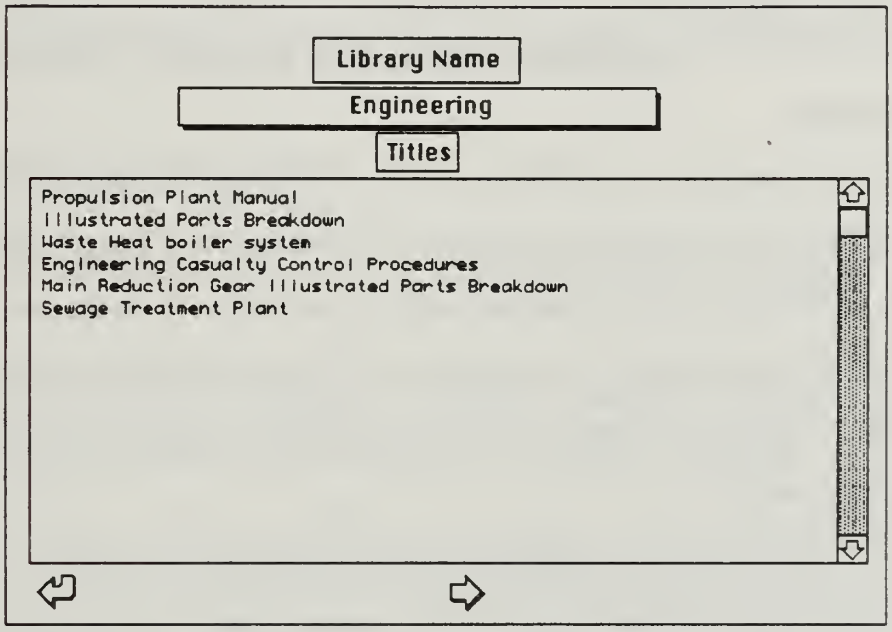


Figure 13. System Manager Window

V. CONCLUSION

A. SUMMARY OF RESEARCH.

TIM is designed to extend the functionality of ARGOS by providing a quick and easy way to search on-line technical information. This means not only supporting the necessary tools, but defining a way to convert current printed material into an electronic format.

Printed material conversion is accomplished through the use of a flat-bed scanner and OCR software. This is sufficient for a demonstration version of TIM, but it is expected that as technology progresses technical documentation will be produced in an electronic format, and will be easily accessible by TIM without the need of a scanner.

Gathering information with TIM involves five steps: Defining the particular library of manuals to be searched, choosing a manual from that library, defining the search key, defining the search type and making a selection from the results of the search. Once the results have been presented, TIM provides three tools to assist in further refining the query. They are: reference to figure linking, auto jump to any page, and string search.

System management is a simple task for shipboard personnel, with only addition and deletion of manuals required on a routine basis. For occasions when a manual must be updated at the shipboard level, a designated system manager can perform the operation with relative ease.

B. REVIEW OF RESULTS

The HyperCard development environment provided many powerful tools that make prototyping easy. However, there are some limitations that make developing a final production version of TIM, in HyperCard, unfeasible at this time. These include:

- HyperTalk is an interpreted, and not a compiled language so the scripts tend to run somewhat slowly.

- Lack of the ability to compile scripts also means that a copy of the HyperCard program must be present to run any stack.

- The current version of HyperCard does not support variable sized windows. This becomes a limitation when working with graphics.

- HyperCard does not support color.

- HyperCard does not have good report generating/printing capabilities.

HyperCard and the Macintosh computer provide a good, easy to use development environment to demonstrate the feasibility of TIM. However, the NeXT® computer, with its UNIX based operating system, would prove to be a more powerful tool to manage some of the difficult functions of ARGOS and TIM.

C. PROPOSED EXTENSIONS AND IMPROVEMENTS

There are several improvements/extensions to TIM that could provide follow on thesis work.

- Redefining the search algorithm in a complied language - A search processing algorithm could be written in a compiled language, like C, as an XCMD.

Working in conjunction with HyperCard's *Find* command, a compiled version of this routine could run faster than the current implementation.

- Interface TIM to the Multi-Backend Database System (MBDS) - Students currently working on the GLAD interface to the MBDS have successfully interfaced a IBM PC compatible to the MBDS through the use of sockets. The same could be done with the Macintosh and HyperCard.

- Produce an efficient Stop/Go List. - Currently there is no way to prevent a search on meaningless words like "and" or "the". There could be a Stop/Go list structure created to determine if the user's query should be processed or rejected.

- Create an indexing of the paragraphs of the manual, by Subject. - If the individual paragraphs of the manual were indexed by subject, then boolen searches for information could be supported. The user could then have the option of switching between text scanning and indexing when constructing a query.

- Port TIM/ARGOS to the NeXT® Computer. - The NeXT® computer has a powerful UNIX based operating system, with a graphical interface. This would make an ideal environment for the development of the next generation of TIM/ARGOS.

- Create a tool to cross reference a query across several manuals - TIM currently requires the user to specify the particular manual to be searched. A tool could be generated that would cross reference a search across all manuals in a particular library.

- Convert TIM to take advantage of HyperCard 2.0 - HyperCard 1.2.2 was used to develop TIM. HyperCard 2.0 is expected to support additional features that would make TIM more efficient. These include: Word Processing fields, text as buttons, variable sized screens, scrolling picture fields and many others.

APPENDIX A

THE VARIOUS BUTTONS AND WHAT THEY DO

This appendix is a listing of the various buttons in TIM and what function each serves:

Return to ARGOS.

Pressing this button will return the user to ARGOS at the point where he left.

Return to the table of contents card.

Located on each tech manual page, this button takes the user back to the table of contents card.

Search.

This icon is located in two different places, but has similar functions in each.

1.) On the opening screen of TIM, when pressed this button creates a customized table of contents card based on the user's search criteria.

2.) On the tech manual page card, when pressed this button prompts the user for a string and then searches the tech manual for the first occurrence, placing a rectangle around it when found. Subsequent occurrences can be found by pressing the return key.

Go to selection.

This button is located on the customized table of contents card. It takes whatever line the user has chosen and goes to that section (page and paragraph number) in the tech manual.

Clear all Cards.

This button clears all cards from the customized table of contents. The results of the next search the user does will go into the first slot of the customized table of contents flip cards.

Delete Current Card.

This button deletes the card that is forward most in the customized table of contents. It then shifts all cards to the left to fill in the space.

Go to System Manager Card.

This opens the section of TIM that allows the system manager to add/delete tech manuals from the various libraries.

Go to Page Number.

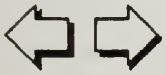
Located on tech manual pages, this button will let the user go to any page number.

Help.

This will provide the user with help on the use of TIM.

Leave ARGOS.

Pressing this button causes the user to leave not only TIM, but ARGOS as well.


Movement Previous and Next.

Used to move around (i.e. Flip Pages) in a tech manual.

Print Page.

This button is located in 2 places:

1.) On the customized table of contents card this button will print out the scrolling field of the card which is upper most.

2.) On the tech manual pages this button will print the entire card that is showing.

Save Current Session to Disk File.

Located on the customized table of contents card, this button will save all of the currently defined searches from the current session to a file. This file can be recalled later to avoid having to do frequently used searches over again.

Customized Table of Contents Tabs.

These tabs are used to "Flip" the customized table of contents cards. The first six characters of the chosen search key are located at the bottom to aid in recalling desired cards.

APPENDIX B

TIM source code

```
SCRIPTS FOR STACK: Tim-3
=================================================

** STACK SCRIPT ************************************

-- When the stack opens, these commands hide the message box, and
-- Show the menu bar

on openStack
  hide message box
  show menuBar
end openStack

-- This function returns the number of the line that is clicked on
-- in a scrolling field.

function choosenLine
  return(trunc(((the mouseV - item 2 of the rect of the target) /¬
  the textheight of the target) + (the scroll of the target / ¬
  textheight of target) + 1))
end choosenLine

-- When the user changes the mode of the search, or the Key, then
-- This function makes the proper adjustments to, and returns, the
-- variable "KeyPhrase".

function SearchChange
  if field SearchType = "Whole Key"
  then
    put "find whole " & quote & field SearchKey & quote &¬
    " in field index" into KeyPhrase
  else if field SearchType = "Contains Key"
  then
    put "find string " & quote & field SearchKey & quote & ¬
    " in field index" into KeyPhrase
  else if field SearchType = "Begins With Key"
  then
    put "find string " & quote && field SearchKey & quote &¬
    " in field index" into KeyPhrase
  else
    put "find string " & quote & field SearchKey && quote &¬
    " in field index" into KeyPhrase
  end if
  put KeyPhrase & "," & char 1 to 3 of field Library & ¬
  field Title & ".idx"into KeyPhrase
```

```
  return KeyPhrase
end SearchChange

--This function takes as input, a line from the tech manual
--index, and formats it to fit into the customized table of contents
--card.

function lineformatter TheLine
  if char 1 of TheLine is "+"
  then
    delete char 1 to 5 of TheLine
    put return before TheLine
  else if char 1 of TheLine is "*"
  then
    put "  " after word 1 of TheLine
    delete char 1 to 5 of TheLine
  else if char 1 of TheLine is "#"
  then
    put "    " after word 1 of TheLine
    delete char 1 to 5 of TheLine
  else
    put "      " after word 1 of TheLine
    delete char 1 to 5 of TheLine
  end if
  return TheLine
end lineformatter

--This is the heart of TIM.  This function takes as input, the KeyWord
--(Which contains the key and the search mode) and the Stack
--of the desired technical manual as input.  It then searches the index
--stack of the manual and returns a formatted variable which contains
--all occurences of the desired key.  Also included is the number of
--times the key was encountered.

function ConductSearch KeyWord,IndexStack
  push card
  set loc of msg to -100,-100
  lock screen
  set cursor to busy
  put 0 into counter
  put empty into SearchList
  put " 0 " into UsedLineNumbers
  put "go to card 1 of stack " & quote & IndexStack & quote into msg
  type return
  doMenu "Find..."
  put KeyWord into msg
  type return
  if the result is empty
  then
    put true into flag
    put (the FoundChunk && id of this card) into StartPoint
    put id of this card into UsedLineFlag
    repeat while flag
      put word 2 of the foundline into HitLineNumber
      if not ((space & HitLineNumber & space) is in UsedLineNumbers)
```

```
      then
        put line HitLineNumber of field Index into holder
        put lineformatter(holder) & return into SearchBlock
        put (space & HitLineNumber & space) after UsedLineNumbers
        add 1 to counter
        put word 1 of (char 2 to 5 of holder) into BackPointer
        repeat while not (space & BackPointer & space is in
          UsedLineNumbers)
          put line BackPointer of field Index into temp
          put space & BackPointer & space after UsedLineNumbers
          put word 1 of (char 2 to 5 of temp) into BackPointer
          put lineformatter(temp) & return before SearchBlock
        end repeat
        put SearchBlock after SearchList
      end if
      type return
      if (the FoundChunk && id of this card) = StartPoint
      then
        put false into flag
      else
        if (id of this card <> UsedLineFlag)
        then
          put id of this card into UsedLineFlag
          put " 0 " into UsedLineNumbers
        end if
      end if
    end repeat
    put counter & space before SearchList
  else
    answer "Key not found." with "OK"
  end if
  hide msg
  set loc of msg to 10,300
  pop card
  unlock screen
  set cursor to hand
  return SearchList
end ConductSearch

--This function is used to sort items in a field.  A bubble sort is very
--inefficient, and was used for demonstration purposes only.

function BubbleSort Thelist
  set cursor to busy
  put number of items of Thelist into counter
  repeat with x = counter down To 1
    repeat with y = x - 1 down To 1
      if item x of Thelist < item y of Thelist
      then
        put item x of TheList into temp
        put item y of Thelist into item x of Thelist
        put temp into item y of Thelist
      end if
    end repeat
  end repeat
```

```
  set cursor to hand
  return Thelist
end BubbleSort

--This handler takes the information returned from the search and
--places it into the proper card on the customized table of contents.
--the field entitled "available" holds an integer (1-9) that represents
--the next available card number in the customized table of contents.
--If the user defines more than 9 searches in one session, than the
--next available card will become number 1.

on SwitchOver
  lock screen
  put false into flag
  go to card Flip1
  put "Flip" & field available into targetcard
  put (field available) mod 9 + 1 into field available
  put field Library of card SearchDefine into field Library¬
  of card value(quote & targetcard & quote)
  put field SearchKey of card SearchDefine into field SearchKey¬
  of card value(quote & targetcard & quote)
  put field SearchType of card SearchDefine into field SearchType¬
  of card value(quote & targetcard & quote)
  put field Title of card SearchDefine into field Title¬
  of card value(quote & targetcard & quote)
  go to card value(quote & targetcard & quote)
  put field SearchKey into holder
  set name of bg button (last char of targetcard) to ¬
  char 1 to 6 of holder
  set scroll of field SearchResults to 0
  put empty into field SearchResults
  unlock screen
end SwitchOver

--This handler goes through the flip cards in the customized table of
--contents and "zeros out" all information.  It also sets the value of
--the field "available" to 1.

on ClearFlipCards
  push card
  lock screen
  repeat with x = 1 to 9
    go to card value(quote & "Flip" & x & quote)
    set name of bg button x to space
    put empty into field Title
    put empty into field SearchTime
    put empty into field Library
    put empty into field SearchKey
    put empty into field SearchType
    put empty into field KeyHits
    put empty into field TargetLine
    set scroll of field SearchResults to 0
    put empty into field SearchResults
  end repeat
  put 1 into field available of card Flip1
```

```
  pop card
  unlock screen
end ClearFlipCards

** BKGND #1, FIELD #1: Library ************************************

-- This creates customized menus for Library Selection.

on mouseDown
  put "Weapons" & return into menu
  put "Operations" & return after menu
  put "Engineering" & return after menu
  put "Navigation" & return after menu
  put "Supply" & return after menu
  Get HPopupMenu(menu,0,The mouseV, The mouseH)
  if not (it is 0)
  then
    put item 1 of it into LineNum
    if LineNum = 1
    then
      put "Weapons" into field Library
    else if LineNum = 2
    then
      put "Operations" into field Library
    else if LineNum = 3
    then
      put "Engineering" into field Library
    else if LineNum = 4
    then
      put "Navigation" into field Library
    else
      put "Supply" into field Library
    end if
    put line 1 of field Titles of card (field Library) into field Title
  end if
end mouseDown

** BKGND #1, FIELD #2: SearchKey ************************************

-- This loads a previous file to the customized table of contents cards.

function LoadFile
  push card
  lock screen
  show field Standby
  unlock screen with visual dissolve
  lock screen
  send ClearFlipCards to this stack
  put FileName ("TEXT") into thefile
  if not (thefile is empty)
  then
    open file thefile
    put empty into it
    read from file thefile until "%"
    if length(it) > 1
```

```
    then
      answer "Illegal file" with "OK"
      close file thefile
    else
      put empty into field SearchKey
      read from file thefile until "•"
      delete last char of it
      put it into field available of card Flip1
      repeat with x = 1 to 9
        go to card value (quote & "Flip" & x & quote)
        read from file thefile until "•"
        if not (it is empty)
        then
          delete last char of it
          put it into field Title
          read from file thefile until "•"
          delete last char of it
          put it into field Library
          read from file thefile until "•"
          delete last char of it
          put it into field SearchKey
          read from file thefile until "•"
          delete last char of it
          put it into field SearchType
          read from file thefile until "•"
          delete last char of it
          put it into field KeyHits
          set scroll of field SearchResults to 0
          read from file thefile until "•"
          delete last char of it
          put it into field SearchResults
          set the name of bg button x to char 1 to 6 of field SearchKey
        else
          exit repeat
        end if
      end repeat
      close file thefile
      pop card
      hide field Standby
      play harpsichord b3 g#4
      unlock screen with dissolve
      go to card Flip1
    end if
  end if
end LoadFile

-- This brings forward the pop-up menu to select search keys.

on mouseDown
  put "Define New" & return & "Load File From Previous Session" & return
into menu
  Get HPopupMenu(menu,0,The mouseV, The mouseH)
  if it is not zero then
    Put Item 1 of it into TheLine
    if TheLine = 1
```

```
    then
      put true into flag
      repeat while flag
        ask "Enter a Key to Search on:" with me
        if not (it is empty)
        then
          if number of words in it = 0
          then
            answer "Enter a Key or Select Cancel" with "OK"
          else
            put it into me
            put false into flag
          end if
        else
          put false into flag
        end if
        set style of field SearchType to rectangle
      end repeat
    else
      put LoadFile() into junk
    end if
  end if
end mouseDown

** BKGND #1, FIELD #3: SearchType ************************************

-- This is gets the user's selection for search type.

on MouseDown
  if not (style of me is "transparent")
  then
    put "Whole Key" & return into menu
    put "Contains Key" & return after menu
    put "Begins With Key" & return after menu
    Put "Ends With Key" & return after Menu
    Get HPopupMenu(menu,0,The mouseV, The mouseH)
    if it is not zero then
      Put Item 1 of it into TheLine
      if TheLine = 1
      then
        put "Whole Key" into field SearchType
      else if TheLine = 2
      then
        put "Contains Key" into field SearchType
      else if TheLine = 3
      then
        put "Begins With Key" into field SearchType
      else
        put "Ends With Key" into field SearchType
      end if
    end if
  end if
end mouseDown

** BKGND #1, FIELD #4: Title ************************************
```

```
-- This brings forward the pop-up menu that displays the available
-- tech manual titles based on the library that is selected.

on mouseDown
  put field Titles of card (field Library) into menu
  Get HPopupMenu(menu,0,The mouseV, The mouseH)
  if not (it is 0)
  then
    put line (item 1 of it) of field Titles of card (field Library)¬
    into me
  end if
end mouseDown


** BKGND #1, BUTTON #1 ************************************

on mouseUp
  play "Boing"
end mouseUp

** BKGND #1, BUTTON #2 ************************************

on mouseUp
  play "Boing"
end mouseUp

** BKGND #1, BUTTON #3 ************************************

on mouseUp
  go to card Supply
end mouseUp

** BKGND #1, BUTTON #4 ************************************

-- When the user quits TIM, the freesize is checked, and if > 10000,
-- the stack is compacted.

on mouseUp
  send ClearFlipCards to this stack
  if (the FreeSize of this stack) > 10000
  then
    doMenu "Compact Stack"
  end if
  put empty into field SearchKey
  pop card
end mouseUp

** BKGND #1, BUTTON #5 ************************************

on mouseUp
  visual scroll left
  go to card Flip1
end mouseUp
```

```
** BKGND #1, BUTTON #6 ************************************

-- This button performs the search, and places the results in the proper
-- card of the customized table of contents.

on mouseUp
  if field SearchKey is empty
  then
    answer "Define a Search Key First." with "OK"
  else
    lock screen
    show field StandBy
    unlock screen with visual dissolve
    put SearchChange () into KeyPhrase
    put the seconds into timer
    put ConductSearch (item 1 of KeyPhrase,item 2 of KeyPhrase)¬
    into temp
    put the seconds - timer into timer
    lock screen
    play harpsichord b3 g#4
    hide field Standby
    unlock screen with visual dissolve
    if not (temp is empty)
    then
      put word 1 of temp into counter
      delete word 1 of temp
      send SwitchOver to this stack
      put timer into field SearchTime
      put counter into field KeyHits
      if number of chars of temp > 29000
      then
        show field SizeWarning
        show bg button OK
        put char 1 to 29000 of temp into field SearchResults
      else
        put temp into field SearchResults
      end if
    end if
  end if
end mouseUp

** BACKGROUND #2: DisplayBack ************************************

-- This deletes exactly one card from the customized table of contents.

on DeleteCard
  set cursor to busy
  put char length(the short name of this card) of ¬
  the short name of this card into holder
  repeat with x = holder to 8
    put "Flip" & x into PresentCard
    put "Flip" & x + 1 into FutureCard
    put field SearchResults of card ¬
    value (quote & FutureCard & quote) into field SearchResults¬
    of card value (quote & PresentCard & quote)
```

```
    put field Title of card ¬
    value (quote & FutureCard & quote) into field Title¬
    of card value (quote & PresentCard & quote)
    put field Library of card ¬
    value (quote & FutureCard & quote) into field Library¬
    of card value (quote & PresentCard & quote)
    put field SearchKey of card ¬
    value (quote & FutureCard & quote) into field SearchKey¬
    of card value (quote & PresentCard & quote)
    put field SearchType of card ¬
    value (quote & FutureCard & quote) into field SearchType¬
    of card value (quote & PresentCard & quote)
    put field KeyHits of card ¬
    value (quote & FutureCard & quote) into field KeyHits¬
    of card value (quote & PresentCard & quote)
    put the name of bg button (x + 1) into temp
    put char (offset(quote,temp) + 1) to (length(temp) - 1) of temp¬
    into temp
    set the name of bg button x to temp
  end repeat
  set scroll of field SearchResults of card Flip9 to 0
  put empty into field SearchResults of card Flip9
  put empty into field Title of card Flip9
  put empty into field Library of card Flip9
  put empty into field SearchKey of card Flip9
  put empty into field SearchType of card Flip9
  put empty into field KeyHits of card Flip9
  set the name of bg button 9 to space
  subtract 1 from field available of card Flip1
  if field available of card Flip1 < 1
  then
    put 9 into field available of card Flip1
  end if
  set cursor to hand
end DeleteCard

** BKGND #2, FIELD #6: SearchResults ************************************

-- For selecting a line from the customized table of contents.

on mouseUp
  set locktext of me to false
  click at the clickloc
  click at the clickloc
  set locktext of me to true
  put word 2 of the selectedline into holder
  put holder & space & value (the selectedline) into field TargetLine
  select line holder of me
end mouseUp

** BKGND #2, BUTTON #1:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
```

```
  go to card id 4467
end mouseUp


** BKGND #2, BUTTON #2:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 7389
end mouseUp


** BKGND #2, BUTTON #3:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 7893
end mouseUp


** BKGND #2, BUTTON #4:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 8377
end mouseUp


** BKGND #2, BUTTON #5:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 8744
end mouseUp


** BKGND #2, BUTTON #6:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 9377
end mouseUp


** BKGND #2, BUTTON #7:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 9837
```

```
end mouseUp


** BKGND #2, BUTTON #8:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 10311
end mouseUp


** BKGND #2, BUTTON #9:   ************************************

on mouseUp
  put empty into field TargetLine
  visual wipe up
  go to card id 10959
end mouseUp


** BKGND #2, BUTTON #10 ************************************

on mouseUp
  put "field SearchResults" into it
  PrintField field SearchResults, the textfont of it,¬
  the textsize of it, the textstyle of it
end mouseUp

** BKGND #2, BUTTON #11 ************************************

on mouseUp
  if field TargetLine is empty
  then
    answer "Make a selection first." with "OK"
  else
    put field TargetLine into holder
    if last char of holder is "•" then delete last char of holder
    put empty into field TargetLine
    put word 1 of holder into linenum
    if not (last char of line linenum of field SearchResults is "•")
    then
      put " •" after line linenum of field SearchResults
    end if
    delete word 1 of holder
    put char 1 to 3 of field Library & field Title & ".txt" ¬
    into TargetStack
    put last word of holder into PageNo
    put "•" & word 1 of holder into ParagraphNo
    push card
    lock screen
    go to card 1 of stack value(quote & TargetStack & quote)
    put GoToPage (PageNo)
    put field PageNumber into temp
   if last char of temp is in "abcdefghij" then delete last char of temp
```

```
    if temp = PageNo then find whole ParagraphNo in field Text
    unlock screen with zoom open
    select the foundChunk
  end if
end mouseUp

** BKGND #2, BUTTON #12 ***********************************

on mouseUp
  visual scroll right
  go to card SearchDefine
end mouseUp

** BKGND #2, BUTTON #13: Delete ***************************************

on mouseUp
  answer "Are you sure you want to delete this card?" with "Yes" or "No"
  if it is "Yes"
  then
    send DeleteCard to bg DisplayBack
  end if
end mouseUp

** BKGND #2, BUTTON #14: OK ************************************

on mouseUp
  hide field SizeWarning
  hide me
end mouseUp

** BKGND #2, BUTTON #15 ************************************

on mouseUp
  answer "Are you sure you want to clear all cards?" with "Yes" or "No"
  if it is "Yes"
  then
    lock screen
    set cursor to busy
    send ClearFlipCards to this stack
    go to card Flip1
    unlock screen
    set cursor to hand
  end if
end mouseUp

** BKGND #2, BUTTON #16 ************************************

on mouseUp
  push card
  lock screen
  if field SearchResults of card Flip1 is empty
  then
    answer "There is nothing to save." with "OK"
  else
    put NewFileName ("Save Current Session As?","") into thefile
```

```
    if not (thefile is empty)
    then
      open file thefile
      write "%" to file thefile
      write field available of card Flipl & "•" to file thefile
      repeat with x = 1 to 9
        put the short name of bg button x into flag
        if flag = space then exit repeat
        go to card value (quote & "Flip" & x & quote)
        write field Title & "•" to file thefile
        write field Library & "•" to file thefile
        write field SearchKey & "•" to file thefile
        write field SearchType & "•" to file thefile
        write field KeyHits & "•" to file thefile
        repeat while (offset("•",field SearchResults)) <> 0
          delete char (offset("•",field SearchResults)) of field ¬
          SearchResults
        end repeat
        write field SearchResults & "•" to file thefile
      end repeat
      close file thefile
      pop card
      unlock screen
      play harpsichord b3 g#4
      answer "Your file has been saved." with "OK"
    end if
  end if
end mouseUp

** BKGND #3, BUTTON #1: New Button ************************************

on mouseUp
  go to card 1
end mouseUp

** CARD #1: SearchDefine ***********************************

on openCard
  hide field Standby
end openCard

** CARD #11, BUTTON #1: New Button *************************************

on mouseUp
  go next
end mouseUp

** CARD #12, BUTTON #1: New Button *************************************

on mouseUp
  go next
end mouseUp

** CARD #12, BUTTON #2: New Button *************************************
```

```
on mouseUp
  go prev
end mouseUp

** CARD #13, BUTTON #1: New Button ************************************

on mouseUp
  go prev
end mouseUp

** CARD #13, BUTTON #2: New Button ************************************

on mouseUp
  go next
end mouseUp

** CARD #14, BUTTON #1: New Button ************************************

on mouseUp
  go next
end mouseUp

** CARD #14, BUTTON #2: New Button ************************************

on mouseUp
  go prev
end mouseUp

** CARD #15, BUTTON #1: New Button ************************************

on mouseUp
  go prev
end mouseUp


••

SCRIPTS FOR STACK: EngPropulsion Plant Manual.idx
=================================================

** STACK SCRIPT ************************************

function choosenLine
  return(trunc(((item 2 of the clickloc - item 2 of the rect¬
  of the target) / the textheight of the target) + 1))
end choosenLine

function checkscroll x,y
  if x > y
  then
    if trunc(x/y) = x/y
    then
      set scroll of field Index to ¬
      (x-y)*the textheight of field Index
    end if
```

```
  end if
end checkscroll

** BACKGROUND #1: IndexBack ************************************

on openStack
  hide message box
  show menuBar
  pass openStack
end openStack

function FixToc LineWidth
  repeat with x = 1 to number of lines of field Index
    put empty into temp
    put line x of field Index into holder
    if not (holder is empty) and not (char 1 of holder is "+")
    then
      repeat with y = 1 to number of chars of holder
        if char y of holder is "."
        then
          put space into char y of holder
        end if
      end repeat
      put word 1 of holder into temp
      delete word 1 of holder
      if (char 1 of temp is in "0123456789") and ("-" is in temp)
      then
        put "." after temp
        repeat until length of temp >= 8
          put space after temp
        end repeat
      else
        put space after temp
        repeat 8 times
          put space before temp
        end repeat
      end if
      repeat with y = 1 to number of words of holder
        put word y of holder & space after temp
      end repeat
      if (char 1 of last word of temp is in "0123456789") and¬
      ("-" is in last word of temp)
      then
        put last word of temp into carrier
        delete last word of temp
        repeat until length of temp >= LineWidth
          put "." after temp
        end repeat
        put space & carrier after temp
      end if
      put temp into line x of field Index
    end if
    put checkscroll (x,10) into junk
  end repeat
  play boing
```

```
  answer "All Done" with "OK"
end FixToc


** BKGND #1, FIELD #3: Error 1 ************************************

on mouseUp
  hide field "Error 1"
end mouseUp

** BKGND #1, FIELD #7 ************************************

function spacer theWord, spaces
  repeat (5 - length(TheWord)) times
    put space after theWord
  end repeat
  return theWord
end spacer

on mouseUp
  put choosenline () into temp
  put line temp of me into holder
  if holder is "Chapter"
  then
    put "+0" into marker
    put field location into field Chaphead
  else if holder is "Top Head"
  then
    put "*" & field ChapHead into marker
    put field location into field TopHead
  else if holder is "Sec Head"
  then
    put "#" & field TopHead into marker
    put field location into field SecHead
  else if holder is "Topic"
  then
    put ">" & field SecHead into marker
    put field location into field Topic
  else if holder is "Previous"
  then
    put "!" into marker
  end if
  put spacer(marker,5) into marker
  put marker before line (field location) of field Index
  put (field location) + 1 into field location
  if number of words of (line (field location) of field Index)¬
  is 0
  then
    repeat until number of words of ¬
      (line (field location) of field Index) > 0
      put (field location) + 1 into field location
    end repeat
  end if
  put checkscroll(field location,8) into junk
end mouseUp
```

```
** BKGND #1, BUTTON #1: Toc Fixer ************************************

on mouseUp
  put FixToc(38) into junk
end mouseUp

** BKGND #1, BUTTON #2: Loi Fixer ************************************

on mouseUp
  put FixToc(70) into junk
end mouseUp

** BKGND #1, BUTTON #3: Indexer **************************************

on mouseUp
  select line (field Location) of field Index
end mouseUp

** BKGND #1, BUTTON #4: Stripper **************************************

on mouseUp
  repeat with x = 1 to number of lines of field Index
    put line x of field Index into holder
    delete char 2 to 4 of holder
    if (char 1 of last word of holder is in "0123456789") and¬
    ("-" is in last word of holder)
    then
      put number of chars of holder into counter
      repeat while counter >= 0
        if char counter of holder is "."
        then
          repeat until not (char counter of holder is ".")
            delete char counter of holder
            put counter - 1 into counter
          end repeat
          exit repeat
        end if
        put counter - 1 into counter
      end repeat
    end if
    put holder into line x of field Index
    put checkscroll(x,8) into junk
  end repeat
  answer "All Done!" with "OK"
end mouseUp

** BKGND #1, BUTTON #5: Numberer **************************************

function spacer TheItem, spaces
  repeat until length(TheItem) >= spaces
    put space after TheItem
  end repeat
  return TheItem
end spacer
```

```
on mouseUp
  put 1 into chaphead
  put 0 into sechead
  put 0 into tophead
  repeat with x = 1 to number of lines of field Index
    put line x of field Index into holder
    if number of words in holder <> 0
    then
      if char 1 of holder is "+"
      then
        put x into chaphead
        put 0 into marker
      else if char 1 of holder is "*"
      then
        put x into tophead
        put chaphead into marker
      else if char 1 of holder is "#"
      then
        put x into sechead
        put tophead into marker
      else
        put sechead into marker
      end if
      put spacer(marker,3) after word 1 of line x of field Index
      if (char 1 of word 2 of line x of field Index is in "01234568789")
      then
        put "." after word 2 of line x of field Index
      end if
    end if
    put checkscroll(x,8) into junk
  end repeat
  play boing
  answer "All Done!" with "OK"
end mouseUp

** BKGND #1, BUTTON #6: Checker ************************************

on mouseUp
  repeat with x = 1 to number of lines of field Index
    put line x of field Index into holder
    if (number of words in holder > 0) and (char 1 of holder <> "+")
    then
      if not ((last char of holder is in "0123456789")¬
      and ("-" is in last word of holder))
      then
        select line x of field Index
        exit repeat
      end if
    end if
    put checkscroll(x,8) into nuk
  end repeat
  play boing
  answer "All Done!" with "OK"
end mouseUp
```

SCRIPTS FOR STACK: EngPropulsion Plant Manual.txt
==========

```
** STACK SCRIPT ************************************

on openStack
  hide message box
  hide menubar
  set loc of msg to -150,-150
end openStack

on closeStack
  hide msg
  set loc of msg to 10,300
  if the freeSize of this stack > 40000
  then
    doMenu "Compact Stack"
  end if
end closeStack

-- Page Locater Using a Binary Search

function GoToPage ThePage
  lock screen
  put 1 into Low
  put number of cards into High
  put char 1 to offset("-",ThePage) - 1 of ThePage into TheBase
  put char offset("-",ThePage) + 1 to length(ThePage) of ThePage¬
  into TheOffset
  put trunc((High + Low)/2) into TheMid
  repeat until Low > High
    put field PageNumber of card TheMid into TestPage
    put "," into char offset("-",TestPage) of TestPage
    if last char of TestPage is in "abcdefghi" then delete last char¬
    of TestPage
    if TheBase < item 1 of TestPage
    then
      put TheMid - 1 into High
    else if TheBase > item 1 of TestPage
    then
      put TheMid + 1 into Low
    else
      if TheOffset < item 2 of TestPage
      then
        put TheMid - 1 into High
      else if TheOffset > item 2 of TestPage
      then
        put TheMid + 1 into Low
      else
        exit repeat
      end if
    end if
```

```
    put trunc((High + Low)/2) into TheMid
  end repeat
  go to card TheMid
  if Low < High
  then
    if last char of field PageNumber is in "abcdefghi"
    then
      repeat until last char of field PageNumber = "a"
        go prev
      end repeat
    end if
  end if
  unlock screen with zoom open
end GoToPage

** BACKGROUND #1 ************************************

on mouseUp
  if not (field Linker is empty)
  then
    put field linker into holder
    put empty into field linker
    visual dissolve
    go to card id holder
  end if
end mouseUp

** BKGND #1, BUTTON #1: Prev ************************************

on mouseUp
  if not (field Linker is empty)
  then
    put empty into field Linker
  end if
  visual effect wipe right
  go to previous card
end mouseUp

** BKGND #1, BUTTON #2 ************************************

on mouseUp
  if (visible of field helper)
  then
    hide field helper
  else
    show field helper
  end if
end mouseUp

** BKGND #1, BUTTON #3: Find It ************************************

on mouseUp
  ask "Find What Page?"
  if not (it is empty)
  then
```

```
    set name of this card to it
    put "nps-cs:The Argos Project:Pete N.:Timƒ:Tech.Docsƒ:Page" & it
into fname
    open file fname
    read from file fname until empty
    put it into field helper
    close file fname
    show field helper
    set scroll of field helper to 0
  end if
end mouseUp

** BKGND #1, BUTTON #4: Number It ************************************

on mouseUp
  repeat with x = 1 to number of lines of field Text
    put line x of field Text into holder
    if (char 1 of holder is in "0123456789") and ("-" is in holder)
    then
      put "•" before line x of field Text
    end if
  end repeat
end mouseUp

** BKGND #1, BUTTON #5: Get New One ************************************

on mouseUp
  put char 1 to (length of field PageNumber - 1) of field PageNumber¬
  into holder
  doMenu "New Card"
  put holder into field PageNumber
  select after line 1 of field PageNumber
end mouseUp

** BKGND #1, BUTTON #6: Next ************************************

on mouseUp
  if not (field Linker is empty)
  then
    put empty into field Linker
  end if
  visual effect wipe left
  go to next card
end mouseUp

** BKGND #1, BUTTON #7: Prev ************************************

on mouseUp
  visual dissolve
  pop card
end mouseUp

** BKGND #1, BUTTON #8: Toggle ************************************

on mouseUp
```

```
  if the optionkey is down
  then
    if (visible of bg button hider)
    then
      hide bg button "But Script Adjust"
      hide bg button hider
      hide bg button Fix
      hide bg button "find it"
      hide bg button "number it"
      hide bg button "Get new one"
      show bg button GoToPage
      show bg button Search
      show bg button Print
    else
      show bg button "But Script Adjust"
      show bg button hider
      show bg button fix
      show bg button "find it"
      show bg button "number it"
      show bg button "Get new one"
      hide bg button Search
      hide bg button Print
      hide bg button GoToPage
    end if
  end if
end mouseUp

** BKGND #1, BUTTON #9: Search ************************************

on mouseUp
  ask "Search For What?" with " "
  if number of words in it > 0
  then
    put "find whole " & quote & it & quote &  " in field Text" into msg
    type return
    if not (the result is empty)
    then
      answer "String Not Found" with "OK"
    end if
  end if
end mouseUp

** BKGND #1, BUTTON #10: Print ************************************

on mouseUp
  play "Boing"
end mouseUp

** BKGND #1, BUTTON #11: Fix **************************************

on mouseUp
  ask "Find find what?" with empty
  if it is not empty
  then
    put it into ToFind
```

```
    ask "Replace With What?" with empty
    if it is not empty then
      put it into ToReplace
      put "find whole " & quote & ToFind & quote into msg
      type return
      repeat while the result is empty
        select the foundchunk
        wait 10
        answer "OK to replace?" with "Yes" or "No" or "Cancel"
        if it is "Yes"
        then
          put ToReplace into the selection
        else if it is "No"
        then
          put value (the foundchunk) into the selection
        else
          exit repeat
        end if
        select after the selection
        put "find whole " & quote & ToFind & quote into msg
        type return
      end repeat
      play harpsichord e
      answer "All done" with "OK"
    end if
  end if
end mouseUp

** BKGND #1, BUTTON #12 ************************************

on mouseUp
  if the optionKey is down
  then
    set loc of msg to 14,356
    show msg
  end if
end mouseUp

** BKGND #1, BUTTON #13: GoToPage ************************************

on mouseUp
  put field PageNumber into holder
  if last char of holder is in "abcdefghijlmnopqrstuvwxyz"
  then
    delete last char of holder
  end if
  ask "Go to which page?" with holder
  if it is not empty and offset("-",it) > 0
  then
    put GoToPage (it) into junk
  end if
end mouseUp

** BKGND #1, BUTTON #14: But Script Adjust ****************************
```

```
on mouseUp
  put "on mouseUp" & return into keep
  put "put the short id of this card into holder" & return after keep
  put "visual dissolve" & return after keep
  put "go to card id xxx" & return after keep
  put "put holder into field Linker" & return after keep
  put "end mouseUp" & return after keep
  repeat with x = 1 to number of cards
    go to card x
    repeat with x = 1 to number of card buttons
      put the script of card button x into temp
      repeat with y = 1 to number of lines of temp
        if "go to card id" is in line y of temp
        then
          put last word of line y of temp into ¬
          last word of line 4 of keep
          set the script of card button x to keep
          exit repeat
        end if
      end repeat
    end repeat
  end repeat
  play harpsichord e
end mouseUp

** BACKGROUND #2 ************************************

on mouseUp
  if not (field Linker is empty)
  then
    put field linker into holder
    put empty into field linker
    visual dissolve
    go to card id holder
  end if
end mouseUp

** BKGND #2, BUTTON #1: Prev ************************************

on mouseUp
  visual dissolve
  pop card
end mouseUp

** BKGND #2, BUTTON #2: Search ************************************

on mouseUp
  ask "Search For What?" with " "
  if number of words in it > 0
  then
    put "find whole " & quote & it & quote &  " in field Text" into msg
    type return
    if not (the result is empty)
    then
      answer "String Not Found" with "OK"
```

```
    end if
  end if
end mouseUp

** BKGND #2, BUTTON #3: Print ************************************

on mouseUp
  doMenu "Print Card"
end mouseUp

** BKGND #2, BUTTON #4 ************************************

on mouseUp
  if the optionKey is down
  then
    set loc of msg to 14,356
    show msg
  end if
end mouseUp

** BKGND #2, BUTTON #5: New Button ************************************

on mouseUp
  put field PageNumber into holder
  if last char of holder is in "abcdefghijlmnopqrstuvwxyz"
  then
    delete last char of holder
  end if
  ask "Go to which page?" with holder
  if it is not empty
  then
    put GoToPage (it) into junk
  end if
end mouseUp

** BKGND #2, BUTTON #6: Prev ************************************

on mouseUp
  if not (field Linker is empty)
  then
    put empty into field Linker
  end if
  visual effect wipe right
  go to previous card
end mouseUp

** BKGND #2, BUTTON #7: Next ************************************

on mouseUp
  if not (field Linker is empty)
  then
    put empty into field Linker
  end if
  visual effect wipe left
  go to next card
```

```
end mouseUp

** CARD #22, BUTTON #1: Table 1-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 9031
  put holder into field Linker
end mouseUp

** CARD #22, BUTTON #2: Table 1-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 9031
  put holder into field Linker
end mouseUp

** CARD #34, BUTTON #1: Figure 2-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 129264
  put holder into field Linker
end mouseUp

** CARD #34, BUTTON #2: Figure 2-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 129264
  put holder into field Linker
end mouseUp

** CARD #41, BUTTON #1: Table 2-1. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 13576
  put holder into field Linker
end mouseUp

** CARD #51, BUTTON #1: Table 2-2 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 139818
  put holder into field Linker
end mouseUp
```

```
** CARD #51, BUTTON #2: Figure 2-16. ***********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 105003
  put holder into field Linker
end mouseUp

** CARD #51, BUTTON #3: Figure 2-26 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 128759
  put holder into field Linker
end mouseUp

** CARD #54, BUTTON #1: Figure 2-18 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 129881
  put holder into field Linker
end mouseUp

** CARD #58, BUTTON #1: Figure 2-19 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 130366
  put holder into field Linker
end mouseUp

** CARD #65, BUTTON #1: Figure 2-26 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 128759
  put holder into field Linker
end mouseUp

** CARD #69, BUTtON #1: Figure 2-26. ***********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 128759
  put holder into field Linker
end mouseUp
```

```
** CARD #73, BUTTON #1: Figure 2-30 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 151941
  put holder into field Linker
end mouseUp

** CARD #103, BUTTON #1: Table 2-3 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 28667
  put holder into field Linker
end mouseUp

** CARD #158, BUTTON #1: Table 2-4 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 42693
  put holder into field Linker
end mouseUp

** CARD #189, BUTTON #1: Table 3-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 40786
  put holder into field Linker
end mouseUp

** CARD #202, BUTTON #1: Table 3-2. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 55254
  put holder into field Linker
end mouseUp

** CARD #203, BUTTON #1: Figure 3-5. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 131646
  put holder into field Linker
end mouseUp

** CARD #208, BUTTON #1: Figure 3-8. ************************************
```

```
on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 158929
  put holder into field Linker
end mouseUp

** CARD #209, BUTTON #1: Table 3-3. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 56636
  put holder into field Linker
end mouseUp

** CARD #223, BUTTON #1: Figure 3-16. **********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 130826
  put holder into field Linker
end mouseUp

** CARD #224, BUTTON #1: Figure 3-20. **********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 132052
  put holder into field Linker
end mouseUp

** CARD #229, BUTTON #1: Figure 3-24. **********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 132447
  put holder into field Linker
end mouseUp

** CARD #232, BUTTON #1: Figure 3-27. **********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 135866
  put holder into field Linker
end mouseUp

** CARD #234, BUTTON #1: Table 3-4 *************************************
```

```
on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 62244
  put holder into field Linker
end mouseUp

** CARD #247, BUTTON #1: Table 4-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 65165
  put holder into field Linker
end mouseUp

** CARD #310, BUTTON #1: Table 5-1. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 82013
  put holder into field Linker
end mouseUp

** CARD #341, BUTTON #1: Table 5-2. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 90684
  put holder into field Linker
end mouseUp

** CARD #345, BUTTON #1: Table 5-3. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 92095
  put holder into field Linker
end mouseUp

** CARD #358, BUTTON #1: Table 5-4. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 95197
  put holder into field Linker
end mouseUp

** CARD #358, BUTTON #2: Table 5-5. ************************************

on mouseUp
```

```
  put the short id of this card into holder
  visual dissolve
  go to card id 95374
  put holder into field Linker
end mouseUp

** CARD #368, BUTTON #1: Table 6-1 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 68983
  put holder into field Linker
end mouseUp

** CARD #371, BUTTON #1: Table 6-2 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 91569
  put holder into field Linker
end mouseUp

** CARD #371, BUTTON #2: Table 6-3 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 98985
  put holder into field Linker
end mouseUp

** CARD #371, BUTTON #3: Figure 6-12. *********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 148061
  put holder into field Linker
end mouseUp

** CARD #377, BUTTON #1: Table 6-4 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 100189
  put holder into field Linker
end mouseUp

** CARD #378, BUTTON #1: Table 6-5 ************************************

on mouseUp
  put the short id of this card into holder
```

```
  visual dissolve
  go to card id 100503
  put holder into field Linker
end mouseUp

** CARD #383, BUTTON #1: Table 6-6. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 101502
  put holder into field Linker
end mouseUp

** CARD #394, BUTTON #1: Table 6-7. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 104348
  put holder into field Linker
end mouseUp

** CARD #399, BUTTON #1: Table 6-8 *************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 96305
  put holder into field Linker
end mouseUp

** CARD #403, BUTTON #1: Table 6-9 *************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 97177
  put holder into field Linker
end mouseUp

** CARD #405, BUTTON #1: Table 6-10. ***********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 98011
  put holder into field Linker
end mouseUp

** CARD #406, BUTTON #1: Table 6-11. ***********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
```

```
  go to card id 98563
  put holder into field Linker
end mouseUp

** CARD #423, BUTTON #1: Table 6-12. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 112159
  put holder into field Linker
end mouseUp

** CARD #424, BUTTON #1: Table 6-13. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 112473
  put holder into field Linker
end mouseUp

** CARD #424, BUTTON #2: Table 6-14. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 113370
  put holder into field Linker
end mouseUp

** CARD #424, BUTTON #3: Figure 6-12 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 148061
  put holder into field Linker
end mouseUp

** CARD #429, BUTTON #1: Table 6-15 *************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 114057
  put holder into field Linker
end mouseUp

** CARD #429, BUTTON #2: Table 6-16 *************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 114317
```

```
  put holder into field Linker
end mouseUp

** CARD #438, BUTTON #1: Table 6-17: ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 116577
  put holder into field Linker
end mouseUp

** CARD #454, BUTTON #1: Table 7-1 **************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 120351
  put holder into field Linker
end mouseUp

** CARD #463, BUTTON #1: Table 7-2 **************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 123228
  put holder into field Linker
end mouseUp

** CARD #466, BUTTON #1: Table 7-3 **************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 123939
  put holder into field Linker
end mouseUp

** CARD #487, BUTTON #1: Table 8-1. *************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 106704
  put holder into field Linker
end mouseUp

** CARD #500, BUTTON #1: Table 8-2 **************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 121106
  put holder into field Linker
```

```
end mouseUp

** CARD #502, BUTTON #1: Table 8-3. ***********************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 133339
  put holder into field Linker
end mouseUp

** CARD #505, BUTTON #1: Table 8-4 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 133977
  put holder into field Linker
end mouseUp

** CARD #509, BUTTON #1: Table 8-5. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 134756
  put holder into field Linker
end mouseUp

** CARD #513, BUTTON #1: Table 8-6 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 136419
  put holder into field Linker
end mouseUp

** CARD #514, BUTTON #1: Table 8-7 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 136663
  put holder into field Linker
end mouseUp

** CARD #525, BUTTON #1: Figure 9-7 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 157639
  put holder into field Linker
end mouseUp
```

```
** CARD #532, BUTTON #1: Table 9-1. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 141094
  put holder into field Linker
end mouseUp

** CARD #533, BUTTON #1: Table 9-2 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 142705
  put holder into field Linker
end mouseUp

** CARD #548, BUTTON #1: Table 9-3 ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 145370
  put holder into field Linker
end mouseUp

** CARD #548, BUTTON #2: Table 9-4. ************************************

on mouseUp
  put the short id of this card into holder
  visual dissolve
  go to card id 146874
  put holder into field Linker
end mouseUp
```

LIST OF REFERENCES

1. Giannotti, G. CDR. and Duffy, Kevin LT. "ARGOS: Design and Development of Object-oriented, Event-driven, Multimedia Database Technology in support of the Paperless Ship", Masters Thesis, Naval Postgraduate School, Monterey, Ca., December 1988.

2. Shafer, D., *HyperTalk Programming*, Indianapolis: Hayden Books, 1988.

3. Webber, J. H., Vice Admiral, USN, Chief Engineer of the Navy, "Naval Engineering and Supply - A Partnership," Address presented at the 1987 Logistics Symposium, Camp Hill, Penn., March 25, 1987.

BIBLIOGRAPHY

Apple Computer, Inc., *Apple: Technical Introduction to the Macintosh Family,* 1987.

Bond, G., *XCMD's for Hypercard*, Management Information Source Inc, 1988.

Goodman, D., *Hypercard™ Developer's Guide*, Bantam Computer Books, 1988.

Kaehler, C., *Hypercard Power - Techniques and Scripts*, Addison-Wesley Publishing Company Inc., 1988.

Knaster, S., *Macintosh Programming Secrets*, Addison-Wesley Publishing Company Inc., 1988.

Korth, H.F., Silberschatz, A., *Database System Concepts*, MacGraw-Hill Book Co., 1986

Swaine, M., *Dr. Dobb's Essential Hypertalk Handbook*, M & T Publishing Inc., 1988.

Shafer, D., *HyperTalk Programming*, Hayden Books, 1988.

Waite Group, *HyperTalk™ Bible*, Hayden Books, 1989.

Waite Group, *Tricks of the HyperTalk™ Masters*, Hayden Books, 1989.

West J., *Programming with the Macintosh Programmer's Workshop*, Bantam Books, 1987.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Dudley Knox Library Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Office of Reasearch Adiministration Code 012 Naval Postgraduate School Monterey, California 93943-5002	1
4.	Chairman, Computer Science Dept. Computer Science Department Naval Postgraduate School Monterey, California 93943-5002	1
5.	Chief of Naval Research 800 N. Quincy Street Arlington, Virginia 22217-5000	1
6.	Center for Naval Analyses 4401 Ford Avenue Arlington, Virginia 22302-0268	1
7.	Naval Ocean Systems Center 271 Catalina Boulevard San Diego, California 92152	1
8.	Curriculum Officer Computer Technology Program, code 37 Monterey, California 93943-5000	1
9.	Professor C. Thomas Wu (Code 52Wq) Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	10

10. Maria M. Jamini-Ramirez 2
Division Head
MDS Division
Data Systems Department
Naval Weapons Station
Concord, California 94520-5000

11. Robert Calogero 2
Director SEA CEL-PA
Logistics Policy and Appraisal Division
Naval Sea Systems Command
Washington, D. C. 20362-5101

12. Clifford G. Geiger 1
Deputy Chief Engineer - Logistics
Naval Sea Systems Command
Washington, D. C. 20362-5101

13. Lt. Peter A. Nardi 3
1133 Mermaid Dr.
Annapolis, Maryland 21401

UTS-587

Thesis
N2452 Nardi
c.1 Development of a hyper-
text oriented technical
information management
system.

Thesis
N2452 Nardi
c.1 Development of a hyper-
text oriented technical
information management
system.

thesN2452
Development of a hypertext oriented tech
3 2768 000 89570 0
DUDLEY KNOX LIBRARY